Python
编程入门指南 上册

明日科技
编著

视频教学+VIP代码库

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

電子工業出版社

VISIT…

内 容 简 介

本书以初学者为对象，全面介绍了使用 Python 进行程序开发的各种技术。在内容排列上由浅入深，让读者循序渐进掌握编程技术；在内容讲解上结合丰富的图解和形象的比喻，帮助读者理解“晦涩难懂”的技术。全书共分 22 章，包括搭建 Python 开发环境、Python 入门必学、变量和数据类型、流程控制语句、序列的应用、字符串及正则表达式、异常处理及程序调试、函数、面向对象程序设计、模块、文件及目录操作、使用 Python 操作数据库、常用的 GUI 框架、Django Web 框架的使用、Pygame 游戏框架、网络爬虫框架、Flask Web 框架的使用、中华诗词大会（网络版）、图像识别工具、微信跳一跳游戏的辅助工具、汽车之家图片抓取工具和巡店宝（企业商品运营利器）。全书内容丰富，项目贴近实际开发，适合读者自学。

本书适用于 Python 的爱好者、初学者和中级开发人员，也可作为大、中专院校和培训机构的教材。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

Python 编程入门指南：全 2 册 / 明日科技编著. —北京：电子工业出版社，2019.5

ISBN 978-7-121-35797-8

Ⅰ. ①P… Ⅱ. ①明… Ⅲ. ①软件工具－程序设计 Ⅳ. ①TP311.561

中国版本图书馆 CIP 数据核字（2018）第 275714 号

策划编辑：徐　岩
责任编辑：张　毅　　特约编辑：田学清
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　　印张：26.5　　字数：679 千字
版　　次：2019 年 5 月第 1 版
印　　次：2019 年 5 月第 1 次印刷
定　　价：80.00 元（全 2 册）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）57565805。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

前 言

Python 是一门有条理的和强大的面向对象的程序设计语言，因其具有丰富和强大的库，还可以把用其他编程语言制作的各种模块（尤其是 C/C++）很轻松地结合在一起，进行高效开发，Python 语言已经成为最受欢迎的程序设计语言之一。Python 语言简单、易学、易用，适合在 Web、数据分析、游戏、科学计算等领域进行编程开发。本书是一本面向 Python 零基础读者的入门书籍，按照基础入门→进阶提升→框架应用→项目开发的循序渐进的过程进行学习和实践。

本书特点

- 结构合理，适合自学。本书定位以初学者为主，在内容安排上充分体现了自学的特点，内容循序渐进，由浅入深，覆盖了编程的各方面技术。
- 视频讲解，全面到位。视频讲解是本书的特色之一，本书提供了近 23 小时的视频同步课程。每个知识点都力求讲透、讲全，在对各知识点的讲解过程中提供了很多编程经验和技巧。
- 比喻生动，容易理解。在编程领域中有许多“晦涩难懂”的名词或技术难以理解，本书为了帮助读者理解，采用了比喻的手法，并结合示意图，帮助读者掌握这些技术。
- 实例丰富，一学就会。书中在介绍知识点的时候，都伴有大量的实例或示例，并且提供实例的设计过程。读者可以按照步骤做出实例，巩固所学的知识。
- 精彩栏目，贴心提醒。本书在各章节中穿插了大量的注意、说明、技巧等栏目，让读者可以在学习过程中更轻松地理解相关知识点及概念，并轻松地掌握个别技术的应用技巧。

读者对象

- 初学编程的自学者。
- 大、中专院校的老师和学生。
- 初、中级程序开发人员。
- 想学习编程的企事业单位在职人员。
- 编程爱好者。
- 相关培训机构的老师和学员。
- 程序测试及维护人员。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

如何研读本书

为了快速掌握本书内容，读者可以按照以下方法研读本书。

- 第一，按顺序阅读。书中的内容是按照由浅入深的顺序精心组织的，对于初学者来说，强烈建议你不要急于学习后面复杂的章节，这样容易顾此失彼，影响学习的自信心。
- 第二，学练结合。对于程序开发人员来说，“理解”和“会做”是两个不同的层次。只有“会做”，也就是按照自己的思维、想法将程序开发出来才能说明完全掌握了知识。
- 第三，多思考。在掌握了一项技术后，要想想该技术还能够解决哪些问题，这样可以使我们掌握的有限的技术发挥最大的功效，在工作中能够充分发挥自己的才能。
- 第四，充满信心。诚然，学习编程是一个艰苦的过程，即使再优秀的图书，也不可能提供所有的经验，解决编程中的所有问题。因为每个人学习编程的经历不同，所积累的经验和遇到的问题不可能完全相同，这就需要我们充满信心，以“不放弃”的态度克服遇到的问题，相信问题终究会解决。

读者服务

为了方便解决本书中的疑难问题，我们提供了多种服务方式，并由作者团队提供在线技术指导和社区服务，服务方式如下。

- 服务网站：www.mingrisoft.com。
- 服务邮箱：mingrisoft@mingrisoft.com。
- 企业 QQ：4006751066。
- QQ 群：539340057。
- 服务电话：400-675-1066、0431-84978981。

致读者

本书由 Python 程序开发团队策划并组织编写，主要编写人员有赛奎春、白兆松、王欢、王小科、冯春龙、王国辉、李磊、杨柳、岳彩龙、潘建羽、葛忠月、李春林、张宝华、杨丽、梁英等。在编写本书的过程中，我们始终坚持“坚韧、创新、博学、笃行”的企业理念，以科学、严谨的态度，力求精益求精，但错误、疏漏之处在所难免，敬请广大读者批评指正。

编　者

2018 年 10 月

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

目 录

第 1 篇 基 础 篇

第 1 章 搭建 Python 开发环境.....1
1.1 Python 概述.....1
1.1.1 了解 Python.....1
1.1.2 Python 的版本.....2
1.1.3 Python 的应用领域.....3
1.2 Python 的下载与安装.....3
1.2.1 你的计算机满足要求吗.....3
1.2.2 下载 Python 安装包.....3
1.2.3 在 Windows 64 位系统上安装 Python.....5
1.2.4 测试 Python 是否安装成功.....7
1.3 Python 开发工具.....7
第 2 章 Python 入门必学.....10
2.1 注释.....10
2.1.1 单行注释.....10
2.1.2 多行注释.....10
2.1.3 中文编码声明注释.....11
2.2 代码缩进.....12
2.3 编码规范.....14
2.3.1 编写规则.....14
2.3.2 命名规范.....15
2.4 基本输入与输出.....15
2.4.1 使用 print()函数输出.....16
2.4.2 使用 input()函数输入.....18

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2.5 快学快用 19
第 3 章 变量和数据类型 21
3.1 保留字与标识符 21
3.1.1 保留字 21
3.1.2 标识符 22
3.2 变量 23
3.2.1 理解 Python 中的变量 23
3.2.2 变量的定义与使用 24
3.3 基本数据类型 25
3.3.1 数字类型 25
3.3.2 字符串类型 27
3.3.3 布尔类型 28
3.3.4 数据类型转换 29
3.4 运算符 29
3.4.1 算术运算符 30
3.4.2 赋值运算符 31
3.4.3 比较（关系）运算符 31
3.4.4 逻辑运算符 32
3.4.5 位运算符 32
3.4.6 运算符的优先级 35
3.5 快学快用 35
第 4 章 流程控制语句 37
4.1 程序结构 37
4.2 选择语句 38
4.2.1 最简单的 if 语句 38
4.2.2 if…else 语句 39
4.2.3 if…elif…else 语句 40
4.2.4 if 语句的嵌套 41
4.3 条件表达式 42
4.4 循环语句 42
4.4.1 while 循环 42
4.4.2 for 循环 43
4.4.3 循环嵌套 45

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

4.5 跳转语句.....46
4.5.1 break 语句.....46
4.5.2 continue 语句.....47
4.6 pass 空语句.....48
4.7 快学快用.....49

第 2 篇 进 阶 篇

第 5 章 序列的应用.....51
5.1 序列.....51
5.1.1 索引（Indexing）.....51
5.1.2 切片（Slicing）.....52
5.1.3 序列相加（Adding）.....53
5.1.4 乘法（Multiplying）.....54
5.1.5 检查某个元素是否是序列的成员（元素）.....54
5.1.6 计算序列的长度、最大值和最小值.....55
5.2 列表（list）.....55
5.2.1 列表的创建和删除.....56
5.2.2 访问列表元素.....58
5.2.3 遍历列表.....58
5.2.4 添加、修改和删除列表元素.....60
5.2.5 对列表进行统计计算.....62
5.2.6 对列表进行排序.....64
5.2.7 列表推导式.....65
5.3 元组（tuple）.....67
5.3.1 元组的创建和删除.....67
5.3.2 访问元组元素.....69
5.3.3 修改元组元素.....70
5.3.4 元组推导式.....71
5.3.5 元组与列表的区别.....72
5.4 字典（dictionary）.....73
5.4.1 字典的创建和删除.....74
5.4.2 通过键–值对访问字典.....77
5.4.3 遍历字典.....78
5.4.4 添加、修改和删除字典元素.....79

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

5.4.5 字典推导式 80
5.5 列表、元组和字典的区别 80
5.6 快学快用 81
第 6 章 字符串及正则表达式 82
6.1 字符串常用操作 82
6.1.1 拼接字符串 82
6.1.2 计算字符串的长度 83
6.1.3 截取字符串 84
6.1.4 分隔字符串 85
6.1.5 检索字符串 86
6.1.6 字母的大小写转换 89
6.1.7 去除字符串中的空格和特殊字符 90
6.1.8 格式化字符串 92
6.2 正则表达式基础 94
6.2.1 行定位符 95
6.2.2 元字符 95
6.2.3 限定符 95
6.2.4 字符类 96
6.2.5 排除字符 96
6.2.6 选择字符 96
6.2.7 转义字符 97
6.2.8 分组 97
6.2.9 在 Python 中使用正则表达式语法 97
6.3 使用 re 模块实现正则表达式操作 98
6.3.1 匹配字符串 98
6.3.2 替换字符串 101
6.3.3 使用正则表达式分隔字符串 102
6.4 快学快用 102
第 7 章 异常处理及程序调试 104
7.1 异常概述 104
7.2 异常处理语句 106
7.2.1 try…except 语句 106
7.2.2 try…except…else 语句 107

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

7.2.3 try…except…finally 语句 108
7.2.4 使用 raise 语句抛出异常 109
7.3 使用 assert 语句调试程序 110
第 8 章 函数 113
8.1 函数的创建和调用 113
8.1.1 创建函数 113
8.1.2 调用函数 115
8.2 参数传递 115
8.2.1 了解形式参数和实际参数 116
8.2.2 位置参数 117
8.2.3 关键字参数 119
8.2.4 为参数设置默认值 119
8.2.5 可变参数 120
8.3 返回值 122
8.4 变量的作用域 122
8.4.1 局部变量 123
8.4.2 全局变量 123
8.5 匿名函数（lambda） 124
8.6 快学快用 125
第 9 章 面向对象程序设计 127
9.1 面向对象概述 127
9.1.1 对象 128
9.1.2 类 128
9.1.3 面向对象程序设计的特点 128
9.2 类的定义和使用 129
9.2.1 定义类 129
9.2.2 创建类的实例 130
9.2.3 创建__init__()方法 130
9.2.4 创建类的成员并访问 132
9.2.5 访问限制 134
9.3 属性（property） 135
9.3.1 创建用于计算的属性 136
9.3.2 为属性添加安全保护机制 137

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

9.4 继承 137
9.4.1 继承的基本语法 138
9.4.2 方法重写 138
9.4.3 派生类中调用基类的__init__()方法 139
9.5 快学快用 140

第 3 篇 高 级 篇

第 10 章 模块 142
10.1 模块概述 142
10.2 自定义模块 143
10.2.1 创建模块 143
10.2.2 使用 import 语句导入模块 144
10.2.3 使用 from…import 语句导入模块 145
10.2.4 模块搜索目录 146
10.3 以主程序的形式执行 147
10.4 Python 中的包 149
10.4.1 Python 程序的包结构 149
10.4.2 创建和使用包 150
10.5 引用其他模块 152
10.6 快学快用 153
第 11 章 文件及目录操作 156
11.1 基本文件操作 156
11.1.1 创建和打开文件 156
11.1.2 关闭文件 159
11.1.3 打开文件时使用 with 语句 159
11.1.4 写入文件内容 159
11.1.5 读取文件 160
11.2 目录操作 163
11.2.1 os 和 os.path 模块 164
11.2.2 路径 165
11.2.3 判断目录是否存在 167
11.2.4 创建目录 167
11.2.5 删除目录 169
11.2.6 遍历目录 170

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

11.3 高级文件操作 171
11.3.1 删除文件 172
11.3.2 重命名文件和目录 172
11.3.3 获取文件基本信息 173
11.4 快学快用 174
第 12 章 使用 Python 操作数据库 176
12.1 数据库编程接口 176
12.1.1 连接对象 176
12.1.2 游标对象 177
12.2 使用 SQLite 178
12.2.1 创建数据库文件 178
12.2.2 操作 SQLite 179
12.3 使用 MySQL 183
12.3.1 下载和安装 MySQL 183
12.3.2 安装 PyMySQL 187
12.3.3 连接数据库 187
12.3.4 创建数据表 188
12.3.5 操作 MySQL 数据表 189

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第1篇　基 础 篇

第 1 章

搭建 Python 开发环境

Python 是一种跨平台的、开源的、免费的、解释型的高级编程语言，近几年发展势头迅猛，在 2018 年 3 月发布的 TIOBE 编程语言排行榜中已经晋升到第 4 名，而在 IEEE Spectrum 发布的 2017 年度编程语言排行榜中，Python 位居第一名。Python 的应用领域非常广泛，例如，Web 编程、图形处理、黑客编程、大数据处理、网络爬虫和科学计算等，Python 都可以实现。

作为 Python 开发的起步，本章先对学习 Python 需要了解的一些基础内容进行简要介绍，然后重点介绍如何搭建 Python 开发环境，最后介绍几种常见的 Python 的开发工具。

1.1　Python概述

1.1.1　了解Python

Python 英文本义是指“蟒蛇”。1989 年，由荷兰人 Guido van Rossum 发明的一种面向对象的解释型高级编程语言，命名为 Python，标志如图 1.1 所示。Python 具有操作简单、开发速度快和容易学习等特点。

图 1.1　Python 的标志

Python 是一种可扩展性强大的编程语言，它具有丰富和强大的库，能够把用其他语言制作的各种模块（尤其是 C/C++）很轻松地结合在一起，所以 Python 常被称为“胶水”语言。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

1991 年，Python 的第一个公开发行版问世。从 2004 年开始，Python 的使用率呈线性增长，逐渐受到编程者的欢迎和喜爱。2010 年，Python 赢得 TIOBE 2010 年度语言大奖；2017 年，在 IEEE Spectrum 发布的 2017 年度编程语言排行榜中，Python 位居第一名，如图 1.2 所示。

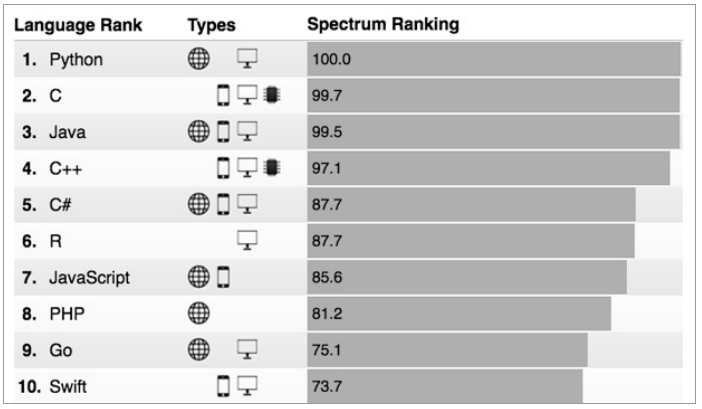


图 1.2　IEEE Spectrum 发布的 2017 年度编程语言排行榜前 10 名

1.1.2　Python的版本

Python 自发布以来，主要经历了三个版本的变化，分别是 1994 年发布的 Python 1.0 版本（已过时）、2000 年发布的 Python 2.0 版本（到 2018 年 3 月已经更新到 2.7.14）和 2008 年发布的 Python 3.0 版本（到 2018 年 3 月已经更新到 3.6.4）。

目前，根据 Semaphore 社区的调查结果显示，使用 Python 2.x 的用户占 63.7%，而使用 Python 3.x 的用户占 36.3%，由此可见，使用 Python 2.x 的用户还是占多数，并且 Python 的作者曾于 2014 年宣布 Python 2.7 支持时间延长到 2020 年。那么作为初学者应该选择什么版本呢？笔者建议初学者选择 Python 3.x 版本。

虽然目前使用 Python 2.x 的用户居多，但是使用 Python 3.x 的用户更愿意进行版本更新，并且使用 Python 3.x 的用户正在迅速扩展，如图 1.3 所示。

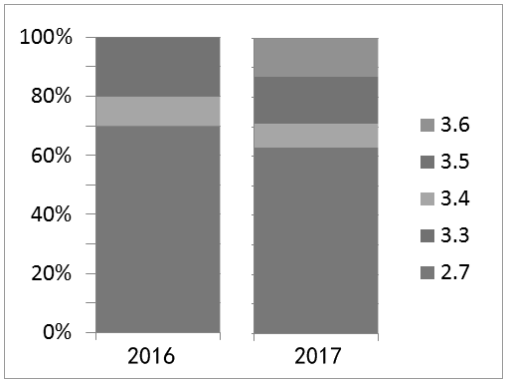


图 1.3　用户使用 Python 版本的对比

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

1.1.3　Python的应用领域

Python 作为一种功能强大且简单易学的编程语言而广受好评，那么 Python 的应用领域有哪些呢？概括起来主要有以下几个方向：

- Web 开发。
- 大数据处理。
- 人工智能。
- 自动化运维开发。
- 云计算。
- 爬虫。
- 游戏开发。

1.2　Python的下载与安装

1.2.1　你的计算机满足要求吗

所谓“工欲善其事，必先利其器”，在正式学习 Python 开发前，需要先搭建 Python 开发环境。由于 Python 是跨平台的，所以可以在多个操作系统上进行编程，并且编写好的程序可以在不同的系统上运行。常用的操作系统及说明如表 1.1 所示。

表 1.1　进行 Python 开发常用的操作系统

操作系统	说　明
Windows	推荐使用 Windows 7 或以上版本。另外，Python 3.5 及以上版本不能在 Windows XP 系统上使用
Mac OS	从 Mac OS X 10.3（Panther）开始已经包含 Python
Linux	推荐 Ubuntu 版本

> **说明：**在个人开发学习阶段推荐使用 Windows 操作系统，本书采用的就是 Windows 操作系统。

1.2.2　下载Python安装包

要进行 Python 开发，需要先安装 Python 解释器。由于 Python 是解释型编程语言，所以需要一个解释器，这样才能运行我们编写的代码。

在 Python 的官方网站中，可以很方便地下载 Python 的开发环境，具体下载步骤如下。

（1）打开浏览器，进入 Python 官方网站，地址是 https:// www.python.org，如图 1.4 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

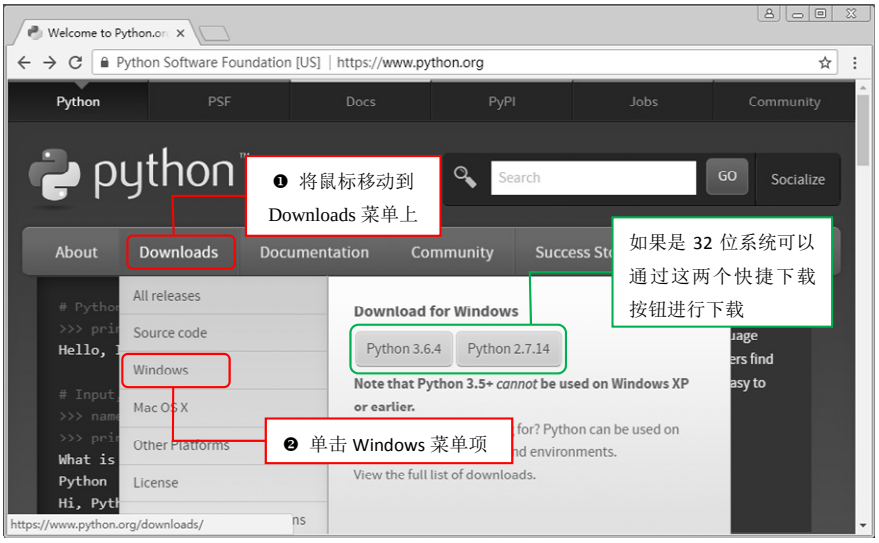


图 1.4　Python 官方网站首页

> **注意：**如果选择 Windows 菜单项时，没有显示右侧的下载按钮，应该是页面没有加载完全，加载完成后就会显示了，请耐心等待。

（2）将鼠标移动到 Downloads 菜单上，将显示和下载有关的菜单项。如果用户使用的是 32 位的 Windows 操作系统，那么直接单击 Python 3.6.4 或者 Python 2.7.14 按钮下载 32 位的安装包，否则，单击 Windows 菜单项，进入详细的下载列表。由于笔者的电脑是 64 位的 Windows 操作系统，所以直接单击 Windows 菜单项，进入如图 1.5 所示的下载列表。

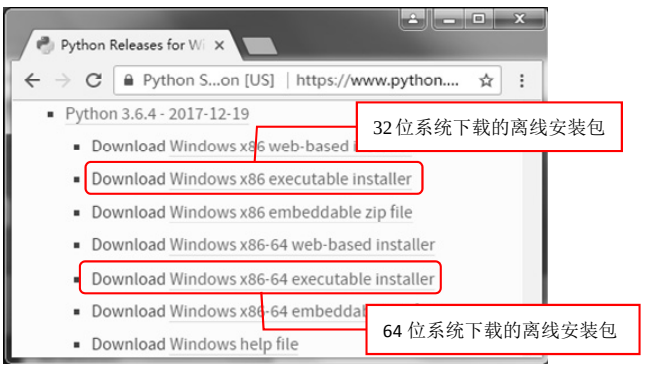


图 1.5　适合 Windows 系统的 Python 下载列表

> **说明：**在如图 1.5 所示的列表中，只带 x86 的，表示是在 Windows 32 位系统上使用的；而带 x86-64 的，则表示是在 Windows 64 位系统上使用的。另外，标记为 web-based installer 的，表示需要连网完成安装；标记为 executable installer 的，表示通过可执行文件（*.exe）方式离线安装；标记为 embeddable zip file 的，表示嵌入式版本，可以集成到其他应用中。

（3）在 Python 下载列表页面中，将列出 Python 提供的各个版本的下载链接。读者可以根

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

据需要下载。当前 Python 3.x 的最新稳定版本是 3.6.4，单击 Windows x86-64 executable installer 超链接，下载适用于 Windows 64 位操作系统的离线安装包，如图 1.6 所示。

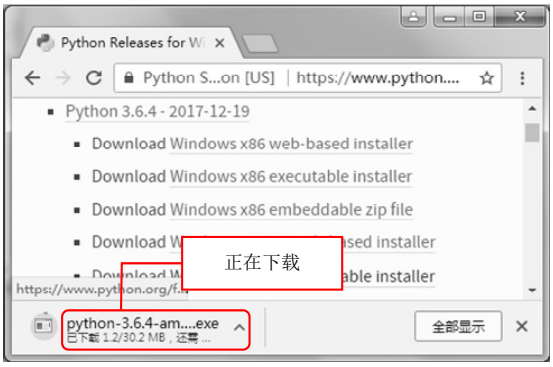


图 1.6　正在下载 Python

（4）下载完成后，浏览器会自动提示“此类型的文件可能会损害您的计算机。您仍然要保留 python-3.6.4-am…exe 吗”，此时单击“保留”按钮，保留该文件即可。

（5）下载完成后，将得到一个名称为 python-3.6.4-amd64.exe 的安装文件。

1.2.3　在Windows 64 位系统上安装Python

在 Windows 64 位系统上安装 Python 3.x 的步骤如下。

（1）双击下载得到的安装文件 python-3.6.4-amd64.exe，将显示安装向导对话框，选中 Add Python 3.6 to PATH 复选框，表示将自动配置环境变量，如图 1.7 所示。

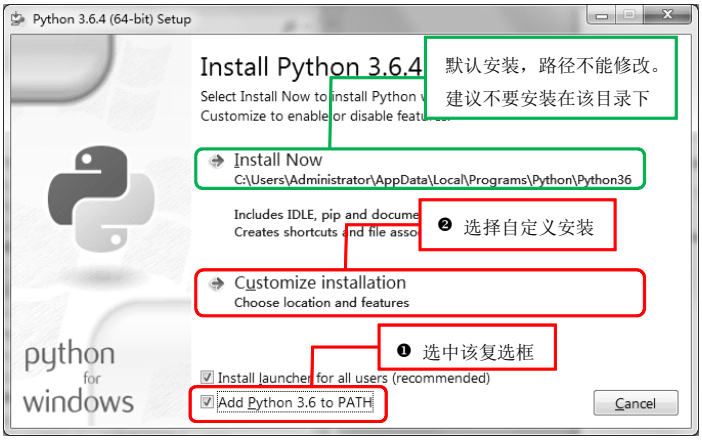


图 1.7　Python 安装向导

（2）单击 Customize installation 按钮，进行自定义安装（自定义安装可以修改安装路径），这里采用默认设置，如图 1.8 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

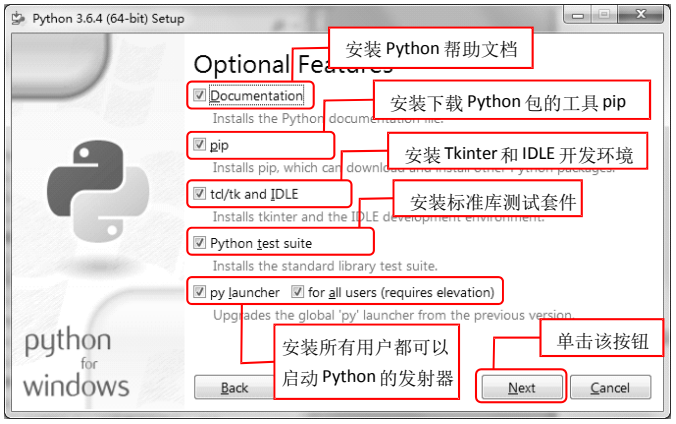


图 1.8　设置安装选项

（3）单击 Next 按钮，在打开的对话框中设置安装路径为 G:\Python\Python36，其他采用默认设置，如图 1.9 所示。

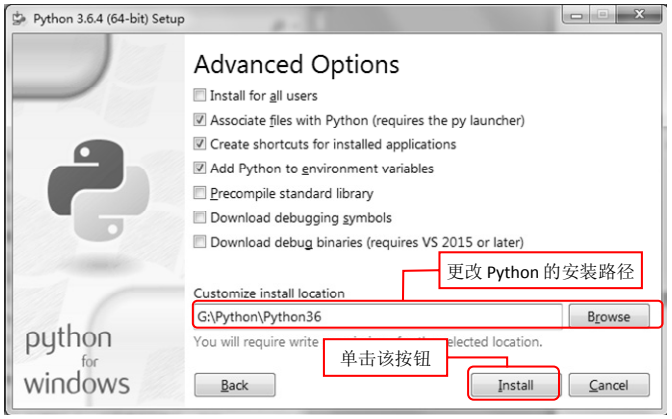


图 1.9　设置安装路径

（4）单击 Install 按钮，开始安装 Python，并且显示安装进度，安装完成后将弹出如图 1.10 所示的对话框。

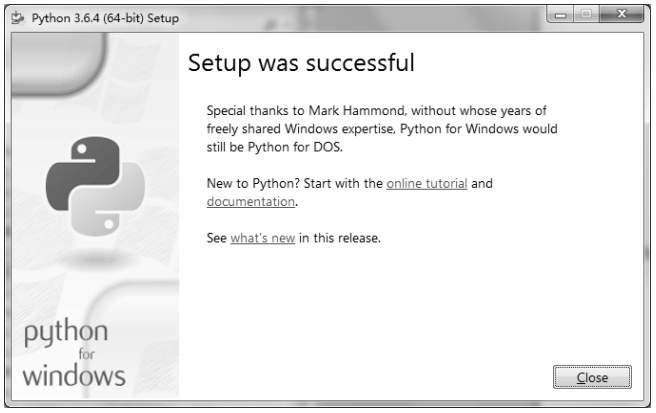


图 1.10　安装完成

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

1.2.4　测试Python是否安装成功

Python 安装完成后，需要检测 Python 是否真的安装成功。例如，在 Windows 7 系统中检测 Python 是否真的安装成功，可以单击 Windows 7 系统的“开始”菜单图标，在“搜索程序和文件”文本框中输入 cmd 命令，然后按〈Enter〉键，启动命令行窗口。在当前的命令提示符后面输入 python，并且按〈Enter〉键，如果出现如图 1.11 所示的信息，则说明 Python 安装成功，同时进入交互式 Python 解释器。

图 1.11　在命令行窗口中运行的 Python 解释器

> **说明：**图 1.11 中的信息是笔者电脑中安装的 Python 的相关信息，其中包括 Python 的版本、该版本发行的时间、安装包的类型等信息。如果读者电脑上的信息与此信息不完全相同也没关系，只要命令提示符变为>>>就说明 Python 已经准备就绪，正在等待用户输入 Python 命令。这也表示我们的 Python 安装成功了。

> **注意：**如果输入 python 后，没有出现如图 1.11 所示的信息，而是显示“‘python’不是内部或外部命令，也不是可运行的程序或批处理文件”，就需要在环境变量中配置 Python。

1.3　Python开发工具

在通常情况下，为了提高开发效率，需要使用相应的开发工具，进行 Python 开发也可以使用开发工具。下面将详细介绍 Python 自带的 IDLE。

在安装 Python 后，会自动安装一个 IDLE，它是一个 Python Shell（可以在打开的 IDLE 窗口的标题栏上看到），也就是一个通过键入文本与程序交互的途径，程序开发人员可以利用 Python Shell 与 Python 交互。下面将详细介绍如何使用 IDLE 开发 Python 程序。

打开 IDLE 时，可以单击 Windows 7 系统的“开始”菜单图标，然后依次选择“所有程序”→“Python 3.6”→“IDLE（Python 3.6 64-bit）”菜单项，打开 IDLE 窗口，如图 1.12 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

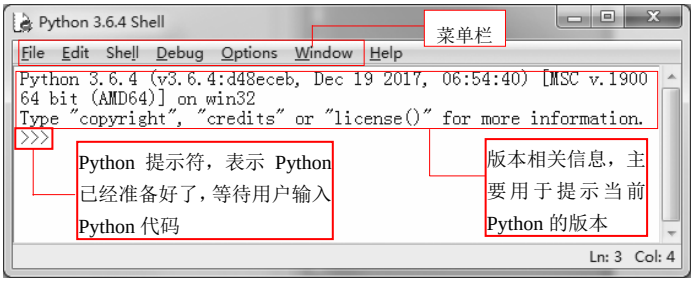


图 1.12　IDLE 主窗口

在实际开发时，通常不会只包含一行代码，如果需要编写多行代码，可以单独创建一个文件保存这些代码，全部编写完毕后再一起执行，具体方法如下。

（1）在 IDLE 主窗口的菜单栏上，选择“File”→“New File”菜单项，在弹出的窗口中可以直接编写 Python 代码，如图 1.13 所示。输入一行代码后按〈Enter〉键，将自动转到下一行，等待继续输入。

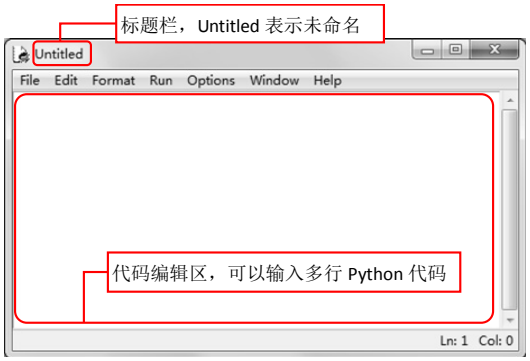


图 1.13　新创建的 Python 文件窗口

（2）在代码编辑区中编写多行代码，代码如下：

```
print(" "*5+"程序员之歌")
print(" "*15+"——《江城子》改编\n")
print("十年生死两茫茫，写程序，到天亮。")
print("千行代码，Bug 何处藏。")
print("纵使上线又怎样，朝令改，夕断肠。")
print("领导每天新想法，天天改，日日忙。")
print("相顾无言，惟有泪千行。")
print("每晚灯火阑珊处，程序员，加班狂。")
```

编写代码后的 Python 文件窗口如图 1.14 所示。

> **注意：**在上面的代码中，小括号()和双引号""都需要在英文半角状态下输入，并且 print 全部为小写字母。Python 的语法是区分字母大小写的。

（3）按下快捷键〈Ctrl+S〉保存文件，这里将其保存为 demo.py，其中.py 是 Python 文件的扩展名。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

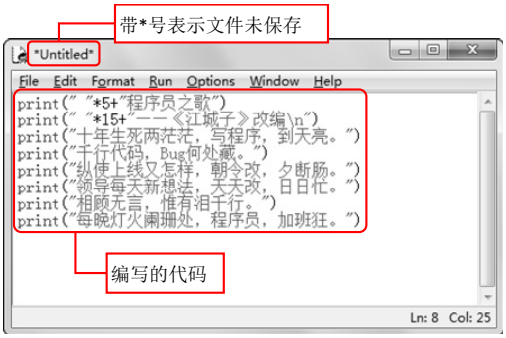


图 1.14　编写代码

（4）运行程序。在菜单栏中选择“Run”→“Run Module”菜单项，如图 1.15 所示。

（5）运行程序后，将弹出 Python Shell 窗口显示运行结果，如图 1.16 所示。

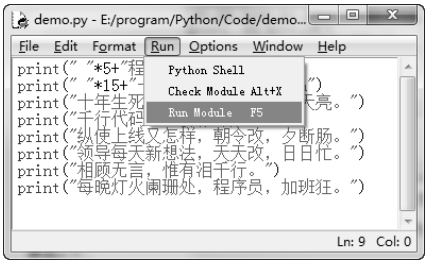


图 1.15　运行程序

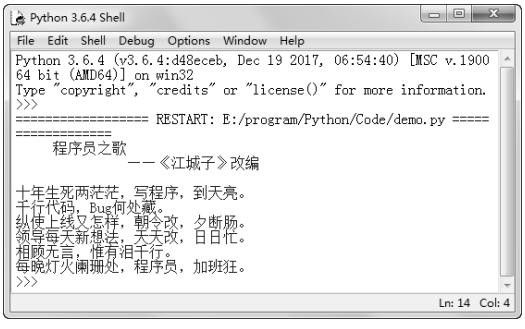


图 1.16　运行结果

说明：可直接按快捷键〈F5〉运行程序。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 2 章

Python 入门必学

2.1 注释

注释，是指在代码中对代码功能进行解释的标注性文字，可以提高代码的可读性。注释的内容将被 Python 解释器忽略，并不会在执行结果中体现出来。

在 Python 中，通常包括 3 种类型的注释，分别是单行注释、多行注释和中文声明注释。

2.1.1 单行注释

在 Python 中，使用“#”作为单行注释的符号。注释从符号“#”开始直到换行为止，其后面所有的内容都作为注释的内容而被 Python 解释器忽略。

语法如下：

```
# 注释内容
```

单行注释可以放在要注释代码的前一行，也可以放在要注释代码的右侧。例如，下面的两种注释形式都是正确的。

第一种形式：

```
# 要求输入出生年份，4 位，如 1983
height=float(input("请输入您的出生年份："))
```

第二种形式：

```
height=float(input("请输入您的出生年份："))      # 要求输入出生年份，4 位，如 1983
```

2.1.2 多行注释

在 Python 中，并没有一个单独的多行注释标记，而是将注释内容包含在一对三引号('''……

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

'''或者"""……"""）之间，这样的代码将被解释器忽略。由于这样的代码可以分为多行编写，所以也可以作为多行注释。

语法格式如下：

```
'''
注释内容 1
注释内容 2
……
'''
```

或者

```
"""
注释内容 1
注释内容 2
……
"""
```

多行注释通常用来为 Python 文件、模块、类或者函数等添加版权、功能等信息，例如，下面的代码将使用多行注释为程序添加功能、开发者、版权、开发日期等信息：

```
'''
信息加密模块
开发者：天星
版权所有：明日科技
2018 年 9 月
'''
```

多行注释也经常用来解释代码中重要的函数、参数等信息，利于后续开发者维护代码，例如：

```
'''
库存类主要的函数方法
update 改/更新
find 查找
delete 删除
create 添加
'''
```

多行注释其实可以采用单行代码多行书写的方式实现，如上面的多行注释可以改写如下：

```
# 库存类主要的函数方法
# update 改/更新
# find 查找
# delete 删除
# create 添加
```

2.1.3　中文编码声明注释

在用 Python 编写代码的时候，如果用到指定字符编码类型的中文编码，需要在文件开头

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

加上中文声明注释，这样可以在程序中有效使用指定字符编码类型的中文编码，不至于出现代码错误，所以说，中文注释很重要。Python 3.x 提供的中文注释声明语法格式如下：

```
# -*- coding:编码 -*-
```

或者

```
# coding=编码
```

例如，保存文件编码格式为 utf-8，可以使用下面的中文编码声明注释：

```
# coding=utf-8
```

一个优秀的程序员，为代码添加注释是必须要做的工作，但要确保注释的内容都是重要的事情，让人看一眼就知道是干什么的；无用的代码是不需要添加注释的。

> **说明：**在上面的代码中，“-*-”没有特殊的作用，只是为了美观才加上的，所以上面的代码也可以使用“# coding:utf-8”代替。

为了让读者透彻理解程序中的代码，本书对大部分代码都进行了注释。在实际开发中，读者只要对关键代码进行注释就可以了。

2.2 代码缩进

Python 不像其他程序设计语言（如 Java 或者 C 语言）采用大括号“{}”分隔代码块，而是采用代码缩进和冒号“:”区分代码之间的层次。

> **说明：**缩进可以使用空格或者<Tab>键实现。使用空格时，通常采用 4 个空格作为一个缩进量；使用<Tab>键时，则采用一个<Tab>键作为一个缩进量。在通常情况下，建议采用空格进行缩进。

在 Python 中，对于类定义、函数定义、流程控制语句，以及异常处理语句等，行尾的冒号和下一行的缩进表示一个代码块的开始，而缩进结束，则表示一个代码块的结束。

例如，下面的代码中的缩进即为正确的缩进：

```
height=float(input("请输入您的身高："))       # 输入身高
weight=float(input("请输入您的体重："))       # 输入体重
bmi=weight/(height*height)                    # 计算 BMI 指数

# 判断身材是否合理
if bmi<18.5:
    print("您的 BMI 指数为："+str(bmi))       # 输出 BMI 指数
    print("体重过轻 ~@_@~")
if bmi>=18.5 and bmi<24.9:
    print("您的 BMI 指数为："+str(bmi))       # 输出 BMI 指数
    print("正常范围，注意保持 (-_-)")
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
if bmi>=24.9 and bmi<29.9:
    print("您的 BMI 指数为: "+str(bmi))        # 输出 BMI 指数
    print("体重过重 ~@_@~")
if bmi>=29.9:
    print("您的 BMI 指数为: "+str(bmi))        # 输出 BMI 指数
    print("肥胖 ^@_@^")
```

Python 对代码的缩进要求非常严格，同一个级别的代码块的缩进量必须相同。如果采用了不合理的代码缩进，将抛出 SyntaxError 异常。例如，代码中有的缩进量是 4 个空格，有的是 3 个空格，就会出现 SyntaxError 错误，如图 2.1 所示。

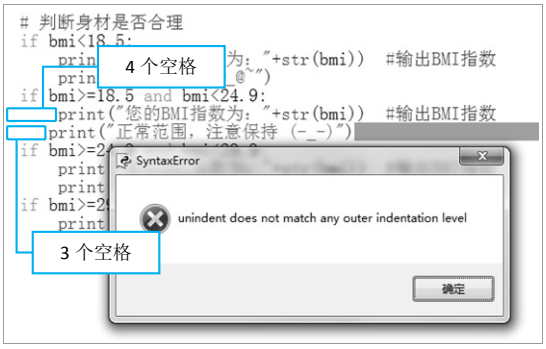


图 2.1　缩进量不同导致的 SyntaxError 错误

在 IDLE 开发环境中，一般以 4 个空格作为基本缩进量。不过也可以选择“Options”→“Configure IDLE”菜单项，在打开的“Settings”对话框的“Fonts/Tabs”选项卡中修改基本缩进量，如图 2.2 所示。

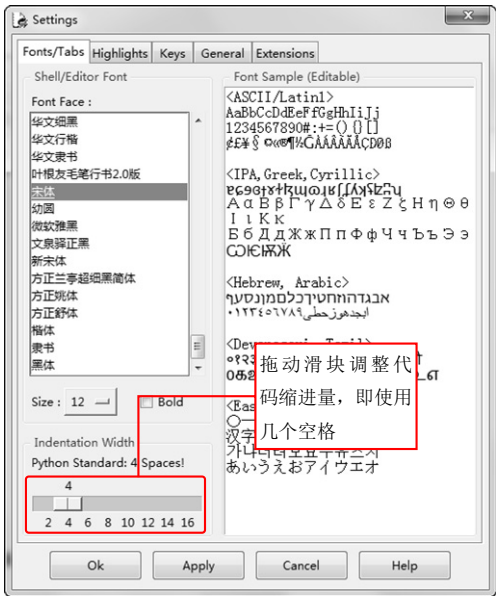


图 2.2　修改基本缩进量

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

多学两招

在 IDLE 开发环境的文件窗口中，可以通过选择主菜单中的“Format”→“Indent Region”菜单项（也可直接使用快捷键〈Ctrl+]〉），将选中的代码缩进（向右移动指定的缩进量），也可通过选择主菜单中的“Format”→“Dedent Region”菜单项（也可直接使用快捷键〈Ctrl+[〉），对代码进行反缩进（向左移动指定的缩进量）。

2.3 编码规范

2.3.1 编写规则

Python 采用 PEP 8 作为编码规范，其中 PEP 是 Python Enhancement Proposal 的缩写，翻译成中文是“Python 增强建议书”，而 8 表示版本，它是 Python 代码的样式指南。下面给出 PEP 8 编码规范中应该严格遵守的条目。

- 每个 import 语句只导入一个模块，尽量避免一次导入多个模块。如图 2.3 所示为推荐写法，如图 2.4 所示为不推荐写法。

```
import os
import sys
```

图 2.3 推荐写法

```
import os, sys
```

图 2.4 不推荐写法

- 不要在行尾添加分号“;”，也不要用分号将两条命令放在同一行。如图 2.5 所示的代码为不规范的写法。

```
height = float(input("请输入您的身高："));
weight = float(input("请输入您的体重："));
```

图 2.5 不规范写法

- 建议每行不超过 80 个字符，如果超过，建议使用小括号“()”将多行内容隐式地连接起来，而不推荐使用反斜杠“\”进行连接。例如，如果一个字符串文本无法在一行中显示，那么可以使用小括号“()”将其分行显示，代码如下：

```
print("我一直认为我是一只蜗牛。我一直在爬，也许还没有爬到金字塔的顶端。"
      "但是只要你在爬，就足以给自己留下令生命感动的日子。")
```

例如，以下通过反斜杠“\”进行连接的做法是不推荐使用的：

```
print("我一直认为我是一只蜗牛。我一直在爬，也许还没有爬到金字塔的顶端。\
但是只要你在爬，就足以给自己留下令生命感动的日子。")
```

不过，以下两种情况除外：

- 导入模块的语句过长。
- 注释里的 URL。

- 使用必要的空行可以提高代码的可读性。一般在顶级定义（如函数或者类的定义）之间

電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

空两行，而在方法定义之间空一行。另外，在用于分隔某些功能的位置也可以空一行。

- 在通常情况下，运算符两侧、函数参数之间、逗号“,”两侧建议使用空格进行分隔。
- 应该避免在循环中使用+和+=运算符累加字符串。这是因为字符串是不可变的，这样做会创建不必要的临时对象。推荐的做法是将每个子字符串加入列表，然后在循环结束后使用 join()方法连接列表。
- 适当使用异常处理结构提高程序容错性，但不能过于依赖异常处理结构，适当的显式判断还是必要的。

2.3.2　命名规范

命名规范在编写代码中起到了很重要的作用，虽然不遵循命名规范程序也可以运行，但是使用命名规范可以更加直观地展示代码所代表的含义。本节将介绍 Python 中常用的一些命名规范。

- 模块名尽量短小，并且全部使用小写字母，可以使用下画线分隔多个字母。例如，game_main、game_register、bmiexponent 都是推荐使用的模块名称。
- 包名尽量短小，并且全部使用小写字母，不推荐使用下画线。例如，com.mingrisoft、com.mr、com.mr.book 都是推荐使用的包名称，而 com_mingrisoft 是不推荐使用的。
- 类名采用单词首字母大写形式，即 Pascal 风格。例如，定义一个借书类，可以命名为 BorrowBook。

> **说明：**Pascal 是为纪念法国数学家 Blaise Pascal 而命名的一种编程语言，Python 中的 Pascal 命名法就是根据该语言的特点总结出来的一种命名方法。

- 模块内部的类采用下画线“_”+Pascal 风格的类名组成。例如，BorrowBook 类的内部类，可以使用_BorrowBook 命名。
- 函数、类的属性和方法的命名规则同模块类似，也是全部采用小写字母，多个字母间用下画线“_”进行分隔。
- 常量命名时全部采用大写字母，可以使用下画线。
- 使用单下画线“_”开头的模块变量或者函数是受保护的，在使用 import * from 语句从模块中导入时这些变量或者函数不能被导入。
- 使用双下画线“__”开头的实例变量或方法是类私有的。

2.4　基本输入与输出

数据的输入与输出操作是计算机最基本的操作。计算机常用的输入与输出设备如图 2.6 所示。本章我们只研究基本的输入与输出，基本输入是指从键盘上输入数据的操作，基本输出是指在屏幕上显示输出结果的操作。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 2.6　常用输入与输出设备

2.4.1　使用print()函数输出

在 Python 中，使用内置的 print()函数可以将结果输出到 IDLE 或者标准控制台上，其基本语法格式如下：

```
print(输出内容)
```

其中，输出内容可以是数字或字符串（使用引号括起来），此类内容将直接输出；也可以是包含运算符的表达式，此类内容将计算结果输出。例如：

```
a = 100                              # 变量 a，值为 100
b = 5                                # 变量 b，值为 5
print(9)                             # 输出数字 9
print(a)                             # 输出变量 a 的值 100
print(a*b)                           # 输出 a*b 的结果 500
print("go big or go home")           # 输出字符串“go big or go home”
```

多学两招

在 Python 中，默认情况下一条 print()语句输出后会自动换行，如果想要一次输出多个内容，而且不换行，可以将要输出的内容使用英文半角的逗号进行分隔。例如下面的代码将在一行中输出变量 a 和 b 的值：

```
print(a,b,“要么出众，要么出局”)      # 输出结果为：100 5 要么出众，要么出局
```

在编程时，我们输入的符号可以使用 ASCII 码的形式输入。ASCII 码是美国信息交换标准码，最早只有 127 个字母被编码到计算机里，也就是大小写英文字母、数字和一些符号，这个编码表被称为 ASCII 编码，比如大写字母 A 的编码是 65，小写字母 z 的编码是 122。通过 ASCII 码显示字符，需要使用 chr()函数进行转换，例如：

```
print("a")                           # 输出字符 a
print(chr（97）)                     # 输出字符 a
print("A")                           # 输出字符 A
print(chr（65）)                     # 输出字符 A
print("B")                           # 输出字符 B
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
print(chr（66）)                    # 输出字符 B
print("+")                          # 输出字符+
print(chr（43）)                    # 输出字符+
print(8)                            # 输出字符 8
print(chr（56）)                    # 输出字符 8
print("[")                          # 输出字符[
print(chr（91）)                    # 输出字符[
print("]")                          # 输出字符]
print(chr（93）)                    # 输出字符]
```

ASCII 码在编程时会经常用到，学习时要掌握 ASCII 码值的一些规律。常用字符与 ASCII 码对照表如图 2.7 所示。

ASCII 非打印字符						ASCII 打印字符											
十进制	字符	代码	十进制	字符	代码	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符	十进制	字符
0	BLANK NULL	NUL	16	▶	DLE	32	(space)	48	0	64	@	80	P	96	`	112	p
1	☺	SOH	17	◀	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	☻	STX	18	↕	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	♥	ETX	19	!!	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	♦	EOT	20	¶	DC4	36	$	52	4	68	D	84	T	100	d	116	t
5	♣	ENQ	21	§	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	♠	ACK	22	▬	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	•	BEL	23	↨	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	◘	BS	24	↑	CAN	40	(	56	8	72	H	88	X	104	h	120	x
9	○	TAB	25	↓	EM	41	)	57	9	73	I	89	Y	105	i	121	y
10	◙	LF	26	→	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	♂	VT	27	←	ESC	43	+	59	;	75	K	91	[	107	k	123	{
12	♀	FF	28	∟	FS	44	,	60	<	76	L	92	\	108	l	124	|
13	♪	CR	29	↔	GS	45	−	61	=	77	M	93	]	109	m	125	}
14	♫	SO	30	▲	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	☼	SI	31	▼	US	47	/	63	?	79	O	95	_	111	o	127	(del)

图 2.7　常用字符与 ASCII 码对照表

随着计算机技术的深入发展，在计算机中不但需要存储和使用基本的英文字符，还需要存储俄语、汉语、日语等文字或符号，随之出现了多种版本的信息转换编码，如 Unicode\utf-8 等。Python 3.0 以 Unicode 为内部字符编码。Unicode 采用双字节 16 位进行编号，可编写 65536 个字符，基本包含了世界上所有的语言字符，成为全世界的一种通用编码，而且用 4 位十六进制值表示一个编码，非常简洁直观，被大多数开发者所接受。打印汉字时可以直接使用 U+编码的形式，如打印汉字“生化危机”和“中国”，代码如下：

```
print ("\u751f\u5316\u5371\u673a")          # 输出字符“生化危机”
print ("\u4e2d\u56fd\")                     # 输出字符“中国”
```

使用 print()函数，不但可以将内容输出到屏幕，还可以输出到指定文件。例如，将字符串

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

“要么出众，要么出局”输出到 D:\mr.txt 中，代码如下：

```
fp = open(r'D:\mr.txt','a+')                    # 打开文件
print("要么出众，要么出局",file=fp)                # 输出到文件中
fp.close()                                      # 关闭文件
```

执行上面的代码后，将在 D:\目录下生成一个名为 mr.txt 的文件，该文件的内容为“要么出众，要么出局”，如图 2.8 所示。

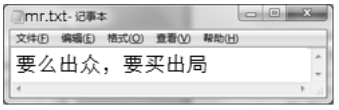


图 2.8　mr.txt 文件的内容

有读者会问：“是否可以将当前年份、月份和日期也输出出来呢？”当然可以，但需要先调用 datetime 模块，并且按指定格式才可以输出相应的日期。例如，要输出当前年份、当前日期和时间，输出代码如下：

```
import datetime                                          # 调用日期模块 datetime
print("当前年份："+datetime.datetime.now().year) # 输出当前年份,当前是 2018 年,输出 2018
# 输出当前日期和时间，如 18-11-20 15:30:23，注意代码中的单引号和字母大小写，不能写错
print("当前日期时间："+datetime.datetime.now（）.dtrftime(‘%y-%m-%d %H:%M:%S’)
```

2.4.2　使用input()函数输入

在 Python 中，使用内置函数 input()可以接收用户的键盘输入。input()函数的基本用法如下：

```
variable = input("提示文字")
```

其中，variable 为保存输入结果的变量，双引号内的文字是用于提示要输入的内容的。例如，想要接收用户输入的内容，并保存到变量 tip 中，可以使用下面的代码：

```
tip = input("请输入文字：")
```

在 Python 3.x 中，无论输入的是数字还是字符，都将被作为字符串读取。如果想要接收数值，需要对接收到的字符串进行类型转换。例如，想要接收整型的数字并保存到变量 num 中，可以使用下面的代码：

```
num = input("请输入您的幸运数字：")
```

前面介绍了使用 ASCII 码值输出相关字符，那么想要获得字符对应的 ASCII 码值该如何实现呢？通过 ord()函数可以将字符的 ASCII 码值转换为数字，下面的代码将实现根据输入的字符输出相应的 ASCII 码值：

```
name=input("输入字符：")                          # 输入字母或数字，不能输入汉字
print(name+" 的 ASCII 码为：",ord(c))              # 显示字符对应的 ASCII 码值
```

如输入字符 A，则输出“A 的 ASCII 码为 81”。输入数字 5，则输出“5 的 ASCII 码为 53”。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

实例 01　**根据输入的年份，计算年龄**

实现根据输入的年份（4 位数字，如 2003）计算目前的年龄，程序中使用 input()函数输入年份、使用 datetime 模块获取当前年份，然后用获取的年份减去输入的年份，就得到了年龄，代码如下：

```
01 import datetime                              # 调入时间模块
02 imyear = input("请输入您的出生年份: ")      # 输入出生年份，必须是 4 位数字，如 2003
03 nowyear= datetime.datetime.now().year) # 计算当前年份
04 age= nowyear- imyear                         # 计算年龄，当前年份减去出生年份
05 print("您的年龄为: "+str(age ) +"岁")      # 输出年龄
06 # 根据计算的年龄判断所处的年龄阶段
07 if age<18:                                   # 如果年龄小于 18 岁
08     print("您现在为未成年人 ~@_@~")          # 输出为“您现在为未成年人 ~@_@~”
09 if age>=18 and age<66:                       # 如果年龄大于 18 岁但小于 66 岁
10     print("您现在为青年人 (-_-)")            # 输出为“您现在为青年人 (-_-)”
11 if age>=66 and age<80:                       # 如果年龄大于或等于 66 岁但小于 80 岁
12     print("您现在为中年人~@_@~")             # 输出为“您现在为中年人~@_@~”
13 if age>=80:                                  # 如果年龄大于或等于 80 岁
14     print("您现在为老年人 *-_-* ")           # 输出为“您现在为老年人*-_-* ”
```

运行程序，提示输入出生年份，如图 2.9 所示。

```
请输入您的出生年份：
```

图 2.9　输入年份提示

输入年份，如输入 2003，按回车键，运行结果如图 2.10 所示。

```
请输入您的出生年份：2003
您的年龄为：15岁
您现在为未成年人 ~@_@~
```

图 2.10　根据输入年份计算年龄

2.5　快学快用

（1）通过本章学到的知识，编写输出程序，看一看这些字符的 ASCII 码值是多少，填入表 2.1 中。

表 2.1　填写 ASCII 码值

字　符	ASCII 值	字　符	ASCII 值	字　符	ASCII 值
@		3		(	
&		0		)	
%		G		?	
\		g		v	

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（2）编写程序，通过字符输出和 ASCII 字符值输出两种方式输出如下表情包。

ASCII 表情	含义	ASCII 表情	含义
:-D	开心	:-(	不悦
:-P	吐舌头	:-*	亲吻
;-)	眨眼	:-x	闭嘴
<※	花束	:-O	惊讶
$_$	见钱眼开	@_@	困惑
>_<	抓狂	T_T	哭泣
= =b	冒冷汗	>3<	亲亲
≧◇≦	感动	= =#	生气
(×_×)	晕倒	(￣^￣)	不满
(=^_^=)	喵喵	(￣¬￣)	流口水
(T_T)	哭泣	ヽ(￣▽￣)ﾉ	两手一摊
╭（╭ _ ╮）╮	路过	(*+﹏+*)~@	受不了
\(^_^)/	为你加油	づ￣ 3￣)づ	飞吻
b（￣▽￣）d	竖起大拇指	(￣(工)￣)	大狗熊
^(oo)^	猪头	Orz	我服了你

（3）Python 提供了很多有用的内置函数或模块，如本章用到的 chr()函数和 ord()函数。通过内置函数可以快速解决开发中遇到的一些问题。接下来每章都提供一些常用的内置函数或模块，大家熟悉一下这些函数或模块的用途，在实际开发时如果遇到相关应用，可以联想到相关函数或模块，然后详细学习并使用，事半功倍。数学运算类内置函数如表 2.2 所示。如果要详细了解这些函数或模块的使用，可以扫描二维码，到快用云端支持平台学习、训练、实战。

表 2.2　数学运算类内置函数（1）

abs(x)	求绝对值 • 参数可以是整型值，也可以是复数 • 若参数是复数，则返回复数的模
oct(x)	将一个数字转换为八进制字符串
hex(x)	将整数 x 转换为十六进制字符串
chr(i)	返回整数 i 对应的 ASCII 字符
ord（x）	返回 x 对应的 ASCII 码值

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 3 章

变量和数据类型

熟练掌握一门编程语言，最好的方法就是充分了解、掌握基础知识，并亲自体验，多敲代码，熟能生巧。

从本章开始，我们将正式踏上 Python 开发之旅，体验 Python 带给我们的简单、快乐。本章主要介绍了 Python 中的保留字、标识符、变量、基本数据类型、数据类型间的转换及运算符。

3.1 保留字与标识符

3.1.1 保留字

保留字是 Python 语言中已经被赋予特定意义的一些单词，在开发程序时，不可以把这些保留字作为变量、函数、类、模块和其他对象的名称来使用。Python 语言中的保留字如表 3.1 所示。

表 3.1 Python 中的保留字

and	as	assert	break	class	continue
def	del	elif	else	except	finally
for	from	False	global	if	import
in	is	lambda	nonlocal	not	None
or	pass	raise	return	try	True
while	with	yield			

> **注意：**Python 中的所有保留字都是区分字母大小写的。例如，if 是保留字，但是 IF 就不属于保留字，如图 3.1 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
>>> true="真"
>>> True="真"
SyntaxError: can't assign to keyword
>>>
```

```
>>> if = "守得云开见月明"
SyntaxError: invalid syntax
>>> IF = "守得云开见月明"
>>>
```

图 3.1　Python 中的保留字区分字母大小写

多学两招

Python 中的保留字可以通过在 IDLE 中输入以下两行代码查看：

```
01 import keyword
02 keyword.kwlist
```

执行结果如图 3.2 所示。

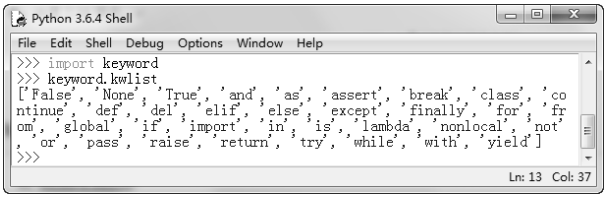


图 3.2　查看 Python 中的保留字

常见错误

如果在开发程序时，使用 Python 中的保留字作为模块、类、函数或者变量等的名称，如下面代码为使用 Python 保留字 if 作为变量的名称：

```
01 if = "坚持下去不是因为我很坚强，而是因为我别无选择"
02 print(if)
```

运行时则会出现如图 3.3 所示的错误提示信息。

图 3.3　使用 Python 保留字作为变量名时的错误信息

3.1.2　标识符

标识符可以简单地理解为一个名字，比如每个人都有自己的名字，它主要用来标识变量、函数、类、模块和其他对象的名称。

Python 语言标识符命名规则如下：

- 由字母、下画线“_”和数字组成，并且第一个字符不能是数字。目前 Python 中只允许使用 ISO-Latin 字符集中的字符 A～Z 和 a～z。
- 不能使用 Python 中的保留字。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

例如，下面是合法的标识符：

```
USERID
name
model2
user_age
```

下面是非法标识符：

```
4word                    # 以数字开头
try                      # Python 中的保留字
$money                   # 不能使用特殊字符$
```

> **注意：**Python 的标识符中不能包含空格、@、%和$等特殊字符。

- 区分字母大小写。

在 Python 中，标识符中的字母是严格区分大小写的，两个同样的单词，如果大小写格式不一样，所代表的意义是完全不同的。例如，下面 3 个变量是完全独立、毫无关系的，就像 3 个长得比较像的人，彼此之间都是独立的。

```
01  number = 0           # 全部小写
02  Number = 1           # 部分大写
03  NUMBER = 2           # 全部大写
```

- Python 中以下画线开头的标识符有特殊意义，一般应避免使用相似的标识符。
  - 以单下画线开头的标识符（如_width）表示不能直接访问的类属性，也不能通过 from xxx import *导入。
  - 以双下画线开头的标识符（如__add）表示类的私有成员。
  - 以双下画线开头和结尾的是 Python 里专用的标识，例如，__init__()表示构造函数。

> **说明：**在 Python 语言中允许使用汉字作为标识符，如“我的名字="明日科技"”，在程序运行时并不会出现错误（如图 3.4 所示），但建议读者尽量不要使用汉字作为标识符。

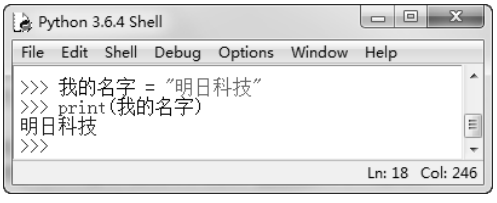


图 3.4　使用汉字作为标识符

3.2　变量

3.2.1　理解Python中的变量

在 Python 中，变量严格意义上应该称为“名字”，也可以理解为标签。当把一个值赋给一

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

个名字时（如把值“学会 Python 还可以飞”赋给 python），python 就称为变量。在大多数编程语言中，都把这称为“把值存储在变量中”，意思是在计算机内存中的某个位置，字符串序列“学会 Python 还可以飞”已经存在。你不需要准确地知道它们到底在哪里，只需要告诉 Python 这个字符串序列的名字是 python，然后就可以通过这个名字来引用这个字符串序列了。

这个过程就像快递员取快递一样，内存就像一个巨大的货架，在 Python 中使用变量就像是给快递盒子贴标签。你的快递存放在货架上，上面贴着写有你名字的标签。当你来取快递时，并不需要知道它们存放在这个大型货架的具体哪个位置，只需要提供你的名字，快递员就会把你的快递交还给你。实际上，你的快递可能并不在原先的位置，不过快递员会为你记录快递的位置。变量也一样，你不需要准确地知道信息存储在内存中的哪个位置，只需要记住存储变量时所用的名字，再使用这个名字就可以了。

3.2.2 变量的定义与使用

在 Python 中，不需要先声明变量名及其类型，直接赋值即可创建各种类型的变量。需要注意的是：对于变量的命名并不是任意的，应遵循以下几条规则：

- 变量名必须是一个有效的标识符。
- 变量名不能使用 Python 中的保留字。
- 慎用小写字母 l 和大写字母 O。
- 应选择有意义的单词作为变量名。

为变量赋值可以通过等号“=”来实现，语法格式如下：

```
变量名 = value;
```

例如，创建一个整型变量，并为其赋值 1024，可以使用下面的语句：

```
number = 1024                          # 创建变量 number 并赋值为 1024，该变量为数值型
```

这样创建的变量就是数值型的变量。如果直接为变量赋值一个字符串值，那么该变量即为字符串类型。例如下面的语句：

```
nickname = "碧海苍梧"                  # 字符串类型的变量
```

另外，Python 是一种动态类型的语言，也就是说，变量的类型可以随时变化。例如，在 IDLE 中，创建变量 nickname，并赋值为字符串"碧海苍梧"，然后输出该变量的类型，可以看到该变量为字符串类型，再将变量赋值为数值 1024，并输出该变量的类型，可以看到该变量为整型。执行过程如下：

```
>>> nickname = "碧海苍梧"          # 字符串类型的变量
>>> print(type(nickname))
<class 'str'>
>>> nickname = 1024               # 整型的变量
>>> print(type(nickname))
<class 'int'>
```

说明：在 Python 语言中，使用内置函数 type()可以返回变量类型。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

在 Python 中，允许多个变量指向同一个值。将两个变量都赋值为数字 2048，再分别应用内置函数 id()获取变量的内存地址，将得到相同的结果。执行过程如下所示：

```
>>> no = 2048
>>> id(no)
49364880
>>> number = 2048
>>> id(number)
49363888
```

> **说明：**在 Python 语言中，使用内置函数 id()可以获取变量的内存地址。

> **注意：**常量就是程序运行过程中值不能改变的量，比如现实生活中的居民身份证号码、数学运算中的π值等，这些都是不会发生改变的，都可以定义为常量。在 Python 中，并没有提供定义常量的保留字，不过在 PEP 8 规范中定义了常量的命名规范为大写字母和下画线组成。但是在实际项目中，常量首次赋值后，还是可以被其他代码修改的。

3.3　基本数据类型

在内存中存储的数据可以有多种类型。例如，一个人的姓名可以用字符类型存储，年龄可以用数值类型存储，而婚否可以用布尔类型存储，这些都是 Python 中的基本数据类型。下面将对这些基本数据类型进行详细介绍。

3.3.1　数字类型

在开发程序时，经常使用数字记录游戏的得分、网站的销售数据和网站的访问量等信息。在 Python 中，提供了数字类型用于保存这些数值，并且它们是不可改变的数据类型。如果修改数字类型变量的值，那么会先把该值存放到内存中，然后修改变量让其指向新的内存地址。

在 Python 中，数字类型主要包括整数、浮点数和复数，下面分别进行介绍。

1．整数

整数用来表示整数数值，即没有小数部分的数值。在 Python 中，整数包括正整数、负整数和 0，并且它的位数是任意的（当超过计算机自身的计算范围时，会自动转用高精度计算），如果要指定一个非常大的整数，只需要写出其所有位数即可。

整数类型包括十进制数、八进制数、十六进制数和二进制数。下面分别进行介绍。

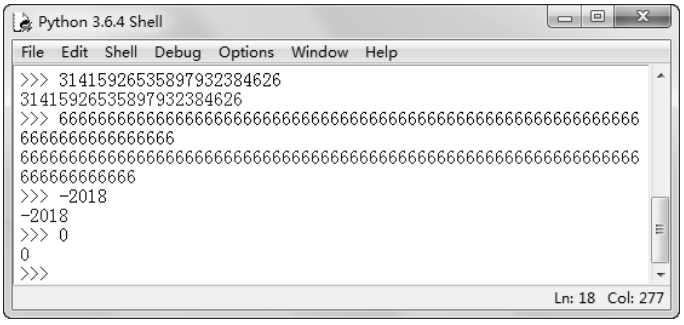
- 十进制数

十进制数的表现形式大家都很熟悉。例如，下面的数值都是有效的十进制数：

```
3141592653589793238462666666666
```

```
-2018
0
```

在 IDLE 中执行的结果如图 3.5 所示。


Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
>>> 3141592653589793238462 6
3141592653589793238462 6
>>> 6666666666

3.1415926535897932384626 等。浮点数也可以使用科学计数法表示，例如，2.7e2、−3.14e5 和 6.16e−2 等。

> **注意：**在使用浮点数进行计算时，可能会出现小数位数不确定的情况。例如，计算 0.1+0.1 时，将得到想要的 0.2，而计算 0.1+0.2 时，将得到 0.30000000000000004（想要的结果为 0.3），执行过程如下所示：

```
>>> 0.1+0.1
0.2
>>> 0.1+0.2
0.30000000000000004
```

> 所有语言都存在这个问题，暂时忽略多余的小数位数即可。

3．复数

Python 中的复数与数学中的复数的形式完全一致，都是由实部和虚部组成的，并且使用 j 或 J 表示虚部。当表示一个复数时，可以将其实部和虚部相加，例如，一个复数，实部为 3.14，虚部为 12.5j，则这个复数为 3.14+12.5j。

3.3.2　字符串类型

字符串就是连续的字符序列，可以是计算机所能表示的一切字符的集合。在 Python 中，字符串属于不可变序列，通常使用单引号“''”、双引号“""”或者三引号“''' '''”“""" """”括起来。这三种引号形式在语义上没有差别，只在形式上有些差别。其中单引号和双引号内的字符序列必须在一行中，而三引号内的字符序列可以分布在连续的多行中。例如，定义 3 个字符串类型变量，并且应用 print()函数输出，代码如下：

```
title = '我喜欢的名言警句'                                    # 使用单引号，字符串内容必须在一行中
mot_cn = "命运给予我们的不是失望之酒，而是机会之杯。" # 使用双引号，字符串内容必须在一行中
# 使用三引号，字符串内容可以分布在多行中
mot_en = '''Our destiny offers not the cup of despair,
but the chance of opportunity.'''
print(title)
print(mot_cn)
print(mot_en)
```

执行结果如图 3.7 所示。

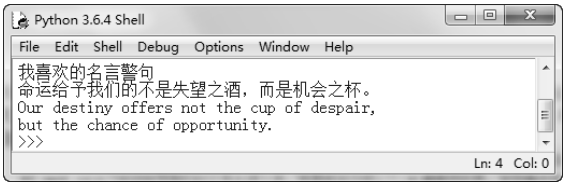


图 3.7　使用 3 种形式定义字符串

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **注意：**字符串开始和结尾使用的引号形式必须一致。另外，当需要表示复杂的字符串时，还可以进行引号的嵌套。例如，下面的字符串也都是合法的：

```
'在 Python 中也可以使用双引号（" "）定义字符串'
"'(‥)nnn'也是字符串"
"""'---' "_"***"""
```

Python 中的字符串还支持转义字符。所谓转义字符是指使用反斜杠“\”对一些特殊字符进行转义。常用的转义字符如表 3.2 所示。

表 3.2　常用的转义字符及其说明

转义字符	说　明
\	续行符
\n	换行符
\0	空
\t	水平制表符，用于横向跳到下一制表位
\"	双引号
\'	单引号
\\	一个反斜杠
\f	换页
\0dd	八进制数，dd 表示字符，如\012 代表换行
\xhh	十六进制数，hh 表示字符，如\x0a 代表换行

> **注意：**在字符串定界符引号的前面加上字母 r（或 R），那么该字符串将原样输出，其中的转义字符将不进行转义。例如，输出字符串“"失望之酒\x0a 机会之杯"”将正常输出，转义字符换行；而输出字符串“r"失望之酒\x0a 机会之杯"”，则原样输出，执行结果如图 3.8 所示。

```
>>> print("失望之酒\x0a机会之杯")
失望之酒
机会之杯
>>> print(r"失望之酒\x0a机会之杯")
失望之酒\x0a机会之杯
>>>
```

图 3.8　转义输出和原样输出的对比

3.3.3　布尔类型

布尔类型主要用来表示真或假的值。在 Python 中，标识符 True 和 False 被解释为布尔值。另外，Python 中的布尔值可以转换为数值，其中 True 表示 1，而 False 表示 0。

> **说明：**Python 中的布尔类型的值可以进行数值运算，例如，“False + 1”的结果为 1，但是不建议对布尔类型的值进行数值运算。

在 Python 中，所有的对象都可以进行真值测试。其中，只有下面列出的几种情况得到的

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

值为假，其他对象在 if 或者 while 语句中都表现为真。

- False 或 None。
- 数值中的零，包括 0、0.0、虚数 0。
- 空序列，包括字符串、空元组、空列表、空字典。
- 自定义对象的实例，对象的__bool__方法返回 False 或者__len__方法返回 0。

3.3.4　数据类型转换

Python 是动态类型的语言（也称为弱类型语言），不需要像 Java 或者 C 语言一样必须在使用变量前声明变量的类型。虽然 Python 不需要先声明变量的类型，但有时仍然需要用到类型转换。例如，在实例 01 中，要想通过一个 print()函数输出提示文字“您的身高:”和浮点型变量 height 的值，就需要将浮点型变量 height 转换为字符串，否则将显示如图 3.9 所示的错误。

```
Traceback (most recent call last):
  File "E:\program\Python\Code\datatype_test.py", line 2, in <module>
    print("您的身高：" + height)
TypeError: must be str, not float
```

图 3.9　字符串和浮点型变量连接时出错

在 Python 中，提供了如表 3.3 所示的函数进行各数据类型间的转换。

表 3.3　常用类型转换函数

函　数	作　用
int(x)	将 x 转换为整数类型
float(x)	将 x 转换为浮点数类型
complex(real [,imag])	创建一个复数
str(x)	将 x 转换为字符串
repr(x)	将 x 转换为表达式字符串
eval(str)	计算在字符串中的有效 Python 表达式，并返回一个对象
chr(x)	将整数 x 转换为一个字符
ord(x)	将一个字符 x 转换为它所对应的整数值
hex(x)	将一个整数 x 转换为一个十六进制字符串
oct(x)	将一个整数 x 转换为一个八进制的字符串

3.4　运算符

运算符是一些特殊的符号，主要用于数学计算、比较大小和逻辑运算等。Python 的运算符主要包括算术运算符、赋值运算符、比较（关系）运算符、逻辑运算符和位运算符。使用运算符将不同类型的数据按照一定的规则连接起来的式子，称为表达式。例如，使用算术运算符连接起来的式子称为算术表达式，使用逻辑运算符连接起来的式子称为逻辑表达式。下面将介绍一些常用的运算符。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

3.4.1 算术运算符

算术运算符是处理四则运算的符号，在数字的处理中应用得最多。常用的算术运算符如表 3.4 所示。

表 3.4 算术运算符

运 算 符	说 明	实 例	结 果
+	加	12.45+15	27.45
-	减	4.56-0.26	4.3
*	乘	5*3.6	18.0
/	除	7/2	3.5
%	求余，即返回除法的余数	7%2	1
//	取整除，即返回商的整数部分	7//2	3
**	幂，即返回 x 的 y 次方	2**4	16，即 2^4

说明：在算术运算符中使用%求余，如果除数（第二个操作数）是负数，那么取得的结果也是一个负值。

注意：使用除法（/或//）运算符和求余运算符时，除数不能为 0，否则将会出现异常，如图 3.10 所示。

```
>>> 5//0
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    5//0
ZeroDivisionError: integer division or modulo by zero
>>> 5/0
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    5/0
ZeroDivisionError: division by zero
>>> 5%0
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    5%0
ZeroDivisionError: integer division or modulo by zero
```

图 3.10 除数为 0 时出现的错误提示

某学员 3 门课程成绩如表 3.5 所示，编程实现：

- Python 课程和 English 课程的分数之差。
- 3 门课程的平均分。

表 3.5 某学员 3 门课程成绩

课 程	分 数
Python	95
English	92
C 语言	89

在 IDLE 中创建一个名称为 score_handle.py 的文件，在该文件中，首先定义 3 个变量，用于存储各门课程的分数，然后应用减法运算符计算分数差，再应用加法运算符和除法运算符计算平均成绩，最后输出计算结果，代码如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
01 python = 95                              # 定义变量，存储 Python 的分数
02 english = 92                             # 定义变量，存储 English 的分数
03 c = 89                                   # 定义变量，存储 C 语言的分数
04 sub = python - english                   # 计算 Python 和 English 的分数差
05 avg = (python + english + c) / 3         # 计算平均成绩
06 print("Python 课程和 English 课程的分数之差： " + str(sub) + " 分\n")
07 print("3 门课的平均分： " + str(avg) + " 分")
```

运行结果如图 3.11 所示。

图 3.11　计算学生成绩的分差及平均分

> **说明：**在 Python 2.x 中，除法运算符（/）的执行结果与 Python 3.x 不一样。在 Python 2.x 中，如果操作数为整数，则结果也将截取为整数，而在 Python 3.x 中，结果为浮点数。例如，7/2，在 Python 2.x 中结果为 3，而在 Python 3.x 中结果为 3.5。

3.4.2　赋值运算符

赋值运算符主要用来为变量等赋值。使用时，可以直接把基本赋值运算符“=”右边的值赋给左边的变量，也可以进行某些运算后再赋值给左边的变量。在 Python 中常用的赋值运算符如表 3.6 所示。

表 3.6　常用的赋值运算符

运 算 符	说　明	举　例	展开形式
=	简单的赋值运算	x=y	x=y
+=	加赋值	x+=y	x=x+y
−=	减赋值	x−=y	x=x−y
*=	乘赋值	x*=y	x=x*y
/=	除赋值	x/=y	x=x/y
%=	取余数赋值	x%=y	x=x%y
**=	幂赋值	x**=y	x=x**y
//=	取整除赋值	x//=y	x=x//y

3.4.3　比较（关系）运算符

比较运算符，也称为关系运算符，用于对变量或表达式的结果进行大小、真假等比较，如果比较结果为真，则返回 True；如果为假，则返回 False。比较运算符通常用在条件语句中作为判断的依据。Python 中的比较运算符如表 3.7 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

表 3.7　Python 中的比较运算符

运 算 符	作　　用	举　　例	结　　果
>	大于	'a' > 'b'	False
<	小于	156 < 456	True
==	等于	'c' == 'c'	True
!=	不等于	'y' != 't'	True
>=	大于等于	479 >= 426	True
<=	小于等于	62.45 <= 45.5	False

多学两招

在 Python 中，当需要判断一个变量是否介于两个值之间时，可以采用“值 1 < 变量 < 值 2”的形式，例如“0 < a < 100”。

3.4.4　逻辑运算符

假定某手机店，在每周二的上午 10 点至 11 点和每周五的下午 14 点至 15 点，对华为 Mate 10 系列手机进行折扣让利活动，那么想参加折扣活动的顾客，就要在时间上满足这样的条件：周二并且 10:00 AM～11:00AM 或者周五并且 2:00 PM～3:00PM。这里就用到了逻辑关系，Python 中也提供了这样的逻辑运算符来进行逻辑运算。

逻辑运算符对真和假两种布尔值进行运算，运算后的结果仍是一个布尔值，Python 中的逻辑运算符主要包括 and（逻辑与）、or（逻辑或）、not（逻辑非）。表 3.8 列出了逻辑运算符的含义和用法。

表 3.8　逻辑运算符

运 算 符	含　　义	用　　法	结合方向
and	逻辑与	op1 and op2	左到右
or	逻辑或	op1 or op2	左到右
not	逻辑非	not op	右到左

使用逻辑运算符进行逻辑运算时，其运算结果如表 3.9 所示。

表 3.9　使用逻辑运算符进行逻辑运算的结果

表达式 1	表达式 2	表达式 1 and 表达式 2	表达式 1 or 表达式 2	not 表达式 1
True	True	True	True	False
True	False	False	True	False
False	False	False	False	True
False	True	False	True	True

3.4.5　位运算符

位运算符用于对二进制数进行计算，因此，需要先将要执行运算的数据转换为二进制数，

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

然后才能进行运算。Python 中的位运算符有位与（&）、位或（|）、位异或（^）、位取反（~）、左移位（<<）和右移位（>>）。

> **说明：**整型数据在内存中以二进制数的形式表示，如整型变量 7 的 32 位二进制数表示为 00000000 00000000 00000000 00000111，其中，左边最高位是符号位，最高位是 0 表示正数，若为 1 则表示负数。负数采用补码表示，如-7 的 32 位二进制数表示为 11111111 11111111 11111111 11111001。

1.“位与”运算

“位与”运算的运算符为“&”，“位与”运算的运算法则是：两个操作数的二进制数表示，只有对应位都是 1 时，结果位才是 1，否则为 0。如果两个操作数的精度不同，则结果的精度与精度高的操作数相同，如图 3.12 所示。

2.“位或”运算

“位或”运算的运算符为“|”，“位或”运算的运算法则是：两个操作数的二进制数表示，只有对应位都是 0，结果位才是 0，否则为 1。如果两个操作数的精度不同，则结果的精度与精度高的操作数相同，如图 3.13 所示。

```
  0000 0000 0000 1100
& 0000 0000 0000 1000
  0000 0000 0000 1000
```

图 3.12　12&8 的运算过程

```
  0000 0000 0000 0100
| 0000 0000 0000 1000
  0000 0000 0000 1100
```

图 3.13　4|8 的运算过程

3.“位异或”运算

“位异或”运算的运算符是“^”，“位异或”运算的运算法则是：当两个操作数的二进制数表示相同（同时为 0 或同时为 1）时，结果为 0，否则为 1。若两个操作数的精度不同，则结果数的精度与精度高的操作数相同，如图 3.14 所示。

4.“位取反”运算

“位取反”运算也称“按位非”运算，运算符为“~”。“取反”运算就是将操作数对应二进制数中的 1 修改为 0，0 修改为 1，如图 3.15 所示。

```
  0000 0000 0001 1111
^ 0000 0000 0001 0110
  0000 0000 0000 1001
```

图 3.14　31^22 的运算过程

```
~ 0000 0000 0111 1011
  1111 1111 1000 0100
```

图 3.15　~123 的运算过程

在 Python 中，使用 print()函数输出如图 3.12～图 3.15 所示的运算结果，代码如下：

```
01 print("12&8 = "+str(12&8))          # 位与计算整数的结果
02 print("4|8 = "+str(4|8))            # 位或计算整数的结果
03 print("31^22 = "+str(31^22))        # 位异或计算整数的结果
04 print("~123 = "+str(~123))          # 位取反计算整数的结果
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

运算结果如图 3.16 所示。

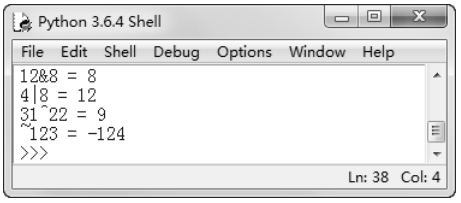


图 3.16　图 3.12～图 3.15 的运算结果

5．左移位运算符<<

左移位运算符<<将一个二进制操作数向左移动指定的位数，左边（高位端）溢出的位被丢弃，右边（低位端）的空位用 0 补充。左移位运算相当于乘以 2 的 *n* 次幂。

例如，int 类型数据 48 对应的二进制数为 00110000，将其左移 1 位，根据左移位运算符的运算规则可以得出(00110000<<1)=01100000，所以转换为十进制数就是 96（48×2）；将其左移 2 位，根据左移位运算符的运算规则可以得出(00110000<<2)=11000000，所以转换为十进制数就是 192（48×2^2），其执行过程如图 3.17 所示。

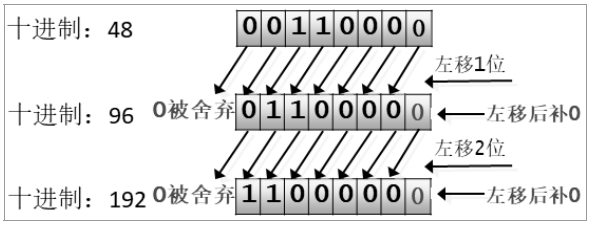


图 3.17　左移位运算

6．右移位运算符>>

右移位运算符>>将一个二进制操作数向右移动指定的位数，右边（低位端）溢出的位被丢弃，而在填充左边（高位端）的空位时，如果最高位是 0（正数），左侧空位填入 0；如果最高位是 1（负数），左侧空位填入 1。右移位运算相当于除以 2 的 *n* 次幂。

正数 48 右移 1 位的运算过程如图 3.18 所示。

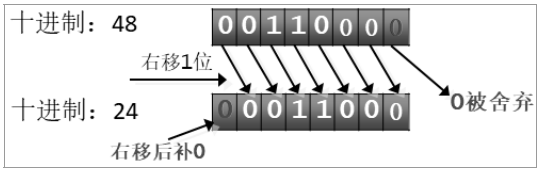


图 3.18　正数的右移位运算过程

负数-80 右移 2 位的运算过程如图 3.19 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

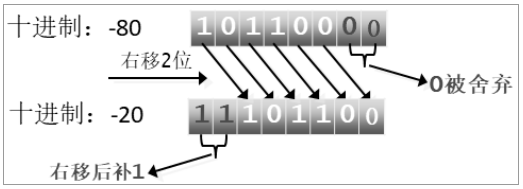


图 3.19　负数的右移位运算过程

多学两招

由于移位运算的速度很快，在程序中遇到表达式乘以或除以 2 的 *n* 次幂的情况时，一般采用移位运算来代替。

3.4.6　运算符的优先级

所谓运算符的优先级，是指在应用中哪一个运算符先计算，哪一个运算符后计算，与数学中的四则运算应遵循的“先乘除，后加减”是一个道理。

Python 运算符的运算规则是：优先级高的运算先执行，优先级低的运算后执行，同一优先级的操作按照从左到右的顺序进行。也可以像四则运算那样使用小括号，括号内的运算最先执行。表 3.10 按从高到低的顺序列出了运算符的优先级。同一行中的运算符具有相同的优先级，此时它们的结合方向决定求值顺序。

表 3.10　运算符的优先级

类　型	说　明
**	幂
~、+、-	取反、正号和负号
*、/、%、//	算术运算符
+、-	算术运算符
<< >>	位运算符中的左移和右移
&	位运算符中的位与
^	位运算符中的位异或
|	位运算符中的位或
<、<=、>、>=、!=、==	比较运算符

多学两招

在编写程序时尽量使用括号“()”来限定运算次序，以免运算次序发生错误。

3.5　快学快用

（1）输出“Go Big Or Go Home”。

世界知名互联网公司都有比较独特的企业文化，如在阿里巴巴，有一个不成文但被严格执

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

行的规定：无论胖瘦、高矮，新进人员都必须在 3 个月内学会靠墙倒立，而且必须坚持 30 秒以上，否则，只能卷铺盖走人。而 Facebook 花了特别多的时间培养新人的野心：如果你没有野心，你就没有办法改变世界。在 Facebook 的办公室中，挂着这样一条充满野心的标语：Go Big Or Go Home（要么成为巨人，要么卷铺盖回家），旁边还配上了哥斯拉的照片，让这条标语显得格外酷炫。下面就输出这条标语：

```
要么成为巨人，要么卷铺盖回家！
                        |
————————————————
| Go Big Or Go Home    |
————————————————
```

（2）大声说出你的爱，敢不敢？啥？不敢！好吧，那就把你的表白写进代码里。请试着定义三个不同类型的变量，输出你的爱之表白吧。

第一个：字符串类型　我爱你一生一世

第二个：浮点型　　520.1314

第三个：整数型　　5211314

（3）Python 提供了很多有用的内置函数或模块，大家熟悉一下这些函数或模块的用途，在实际开发时如果遇到相关应用，可以联想到相关函数或模块，然后详细学习并使用，事半功倍。数学运算类内置函数如表 3.11 所示。如果要详细了解这些函数或模块的使用，可以扫描二维码，到快用云端支持平台学习、训练、实战。

表 3.11　数学运算类内置函数（2）

complex([real[, imag]])	创建一个复数
divmod(a, b)	分别取商和余数 **注意：**整型、浮点型都可以
float([x])	将一个字符串或数转换为浮点数。如果无参数将返回 0.0
int([x[, base]])	将一个字符转换为 int 类型，base 表示进制
long([x[, base]])	将一个字符转换为 long 类型

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 4 章

流程控制语句

做任何事情都要遵循一定的原则。例如，到图书馆去借书，就必须要有借书证，并且借书证不能过期，这两个条件缺一不可。程序设计也是如此，需要利用流程控制实现与用户的交流，并根据用户的需求决定程序“做什么”“怎么做”。

流程控制对于任何一门编程语言来说都是至关重要的，它提供了控制程序如何执行的方法。如果没有流程控制语句，整个程序将按照线性顺序来执行，而不能根据用户的需求决定程序执行的顺序。本章将对 Python 中的流程控制语句进行详细讲解。

4.1 程序结构

计算机在解决某个具体问题时，主要有 3 种情形，分别是顺序执行所有的语句、选择执行部分语句和循环执行部分语句。对应程序设计中的 3 种基本结构是顺序结构、选择结构和循环结构。这 3 种结构的执行流程如图 4.1 所示。

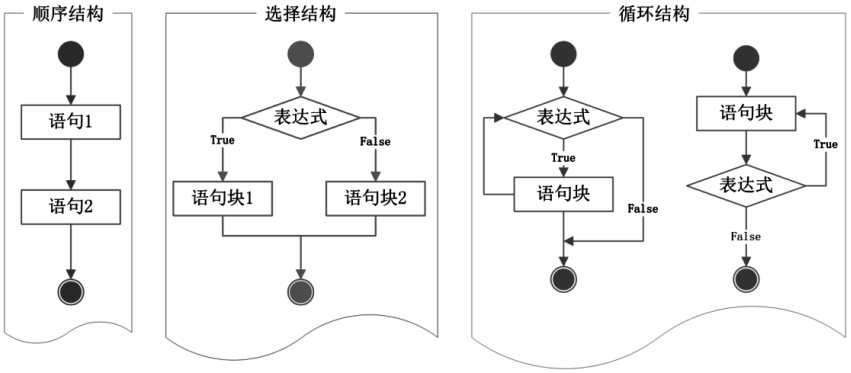


图 4.1　结构化程序设计的 3 种基本结构

其中，第一幅图是顺序结构的流程图，编写完毕的语句按照编写顺序依次被执行；第二幅

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图是选择结构的流程图，它主要根据条件语句的结果选择执行不同的语句；第三幅图是循环结构的流程图，它是在一定条件下反复执行某段程序的流程结构，其中，被反复执行的语句称为循环体，而决定循环是否中止的判断条件称为循环条件。

本章之前编写的多数例子采用的都是顺序结构。例如，定义一个字符串类型的变量，然后输出该变量，代码如下：

```
01 mot_cn = "命运给予我们的不是失望之酒，而是机会之杯。"  # 使用双引号，字符串内容必须在一行
02 print(mot_cn)
```

选择结构和循环结构的应用场景：

看过《射雕英雄传》的人，可能会记得，黄蓉与瑛姑见面时，曾出过这样一道数学题：今有物不知其数，三三数之剩二，五五数之剩三，七七数之余二，问几何？

解决这道题，有以下两个要素：

- 需要满足的条件是一个数，除以三余二，除以五余三，除以七余二。这就涉及条件判断，需要通过选择语句实现。
- 依次尝试符合条件的数。这就需要循环执行，需要通过循环语句实现。

4.2 选择语句

在生活中，我们总是要做出许多选择，程序也是一样。下面给出几个常见的例子：

- 如果购买商品成功，用户余额减少，用户积分增多。
- 如果输入的用户名和密码正确，提示登录成功，进入网站，否则，提示登录失败。
- 如果用户使用微信登录，则使用微信扫一扫；如果使用 QQ 登录，则输入 QQ 号和密码；如果使用微博登录，则输入微博账号和密码；如果使用手机号登录，则输入手机号和密码。

以上例子中的判断，就是程序中的选择语句，也称为条件语句，即按照条件选择执行不同的代码片段。Python 中选择语句主要有 3 种形式，分别为 if 语句、if…else 语句和 if…elif…else 多分支语句，下面将分别对它们进行详细讲解。

4.2.1 最简单的if语句

Python 中使用 if 保留字来组成选择语句，其最简单的语法形式如下：

```
if 表达式:
    语句块
```

其中，表达式可以是一个单纯的布尔值或变量，也可以是比较表达式或逻辑表达式（例如，a > b and a != c），如果表达式的值为真，则执行“语句块”；如果表达式的值为假，就跳过“语句块”，继续执行后面的语句。这种形式的 if 语句相当于汉语里的“如果……就……”，其流程图如图 4.2 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

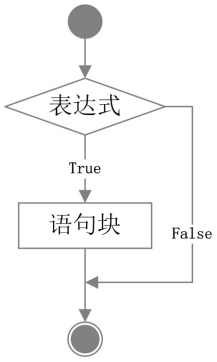


图 4.2　最简单 if 语句的执行流程

> **说明：**在 Python 中，当表达式的值为非零的数或者非空的字符串时，if 语句也认为是条件成立（为真值）。

4.2.2　if…else语句

如果遇到只能二选一的条件，Python 中提供了 if…else 语句解决类似问题，其语法格式如下：

```
if 表达式:
        语句块 1
else:
        语句块 2
```

使用 if…else 语句时，表达式可以是一个单纯的布尔值或变量，也可以是比较表达式或逻辑表达式，如果满足条件，则执行 if 后面的语句块，否则，执行 else 后面的语句块。这种形式的选择语句相当于汉语里的“如果……否则……”，其流程图如图 4.3 所示。

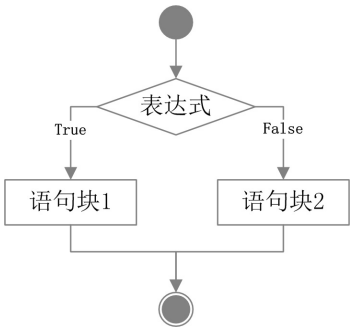


图 4.3　if…else 语句流程图

> **技巧：**if…else 语句可以使用条件表达式进行简化，如下面的代码：

```
01  a = -9
02  if a > 0:
03      b = a
04  else:
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
05      b = -a
06  print(b)
```

可以简写成：

```
01  a = -9
02  b = a if a>0 else -a
03  print(b)
```

上段代码主要实现求绝对值的功能，如果 a>0，就把 a 的值赋值给变量 b，否则将-a 赋值给变量 b。使用条件表达式的好处是可以使代码简洁，并且有一个返回值。

4.2.3 if…elif…else语句

大家平时在网上购物时，通常都有多种付款方式以供选择，如图 4.4 所示。

图 4.4　购物时的付款页面

图 4.4 中提供了 3 种付款方式，这时用户就需要从多个选项中选择一个。在开发程序时，如果遇到多选一的情况，则可以使用 if…elif…else 语句，该语句是一个多分支选择语句，通常表现为“如果满足某种条件，进行某种处理，否则，如果满足另一种条件，则执行另一种处理……”。if…elif…else 语句的语法格式如下：

```
if 表达式1:
    语句块1
elif 表达式2:
    语句块2
elif 表达式3:
    语句块3
…
else:
    语句块n
```

使用 if…elif…else 语句时，表达式可以是一个单纯的布尔值或变量，也可以是比较表达式或逻辑表达式，如果表达式为真，执行语句；如果表达式为假，则跳过该语句，进行下一个 elif 的判断，只有在所有表达式都为假的情况下，才会执行 else 中的语句。if…elif…else 语句

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

的流程图如图 4.5 所示。

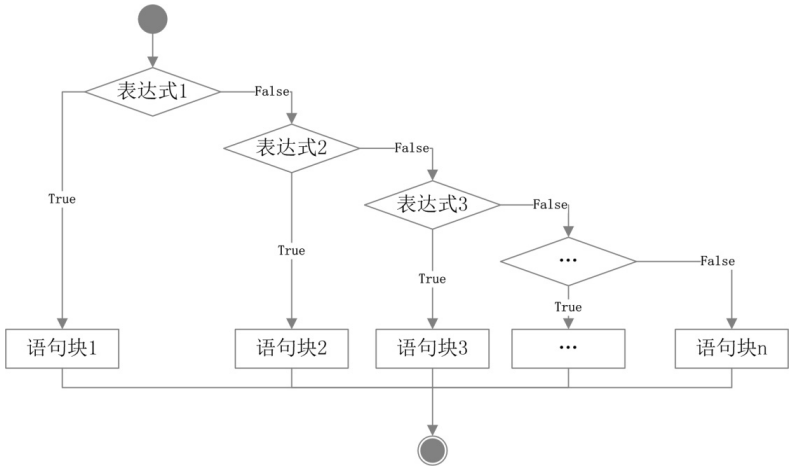


图 4.5　if…elif…else 语句的流程图

> **注意：**if 和 elif 都需要判断表达式的真假，而 else 则不需要判断；另外，elif 和 else 都必须与 if 一起使用，不能单独使用。

4.2.4　if语句的嵌套

前面介绍了 3 种形式的 if 选择语句，这 3 种形式的选择语句之间都可以进行互相嵌套。例如，在最简单的 if 语句中嵌套 if…else 语句，形式如下：

```
if 表达式1:
    if 表达式2:
        语句块1
    else:
        语句块2
```

例如，在 if…else 语句中嵌套 if…else 语句，形式如下：

```
if 表达式1:
    if 表达式2:
        语句块1
    else:
        语句块2
else:
    if 表达式3:
        语句块3
    else:
        语句块4
```

> **说明：**if 选择语句可以有多种嵌套方式，在开发程序时，可以根据自身需要选择合适的嵌套方式，但一定要严格控制好不同级别代码块的缩进量。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

4.3 条件表达式

在程序开发时，经常会根据表达式的结果，有条件地进行赋值。例如，要返回两个数中较大的数，可以使用下面的 if 语句：

```
01 a = 10
02 b = 6
03 if a>b:
04     r = a
05 else:
06     r = b
```

针对上面的代码，可以使用条件表达式进行简化，代码如下：

```
01 a = 10
02 b = 6
03 r = a if a > b else b
```

在使用条件表达式时，先计算中间的条件（a>b），如果结果为 True，返回 if 语句左边的值，否则返回 else 右边的值。例如上面表达式的结果，即 r 的值为 10。

> **说明：**Python 中提供的条件表达式，可以根据表达式的结果进行有条件的赋值。

4.4 循环语句

在日常生活中很多问题都无法一次解决，如盖楼，所有高楼都是一层一层垒起来的。有些事物必须周而复始地运转才能保证其存在的意义，例如，公交车、地铁等交通工具必须每天在同样的时间往返于始发站和终点站之间。类似这样的反复做同一件事的情况，称为循环。循环主要有两种类型：

- 重复一定次数的循环，称为计次循环，如 for 循环。
- 一直重复，直到条件不满足时才结束的循环，称为条件循环。只要条件为真，这种循环会一直持续下去，如 while 循环。

下面将对这两种类型的循环分别进行介绍。

> **说明：**在其他语言中（例如，C、C++、Java 等），条件循环还包括 do…while 循环，但是，在 Python 中没有 do…while 循环。

4.4.1 while循环

while 循环是通过一个条件来控制是否要继续反复执行循环体中的语句。语法如下：

```
while 条件表达式:
    循环体
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

说明：循环体是指一组被重复执行的语句。

当条件表达式的返回值为真时，则执行循环体中的语句，执行完毕后，重新判断条件表达式的返回值，直到表达式返回的结果为假时，退出循环。while 循环语句的执行流程如图 4.6 所示。

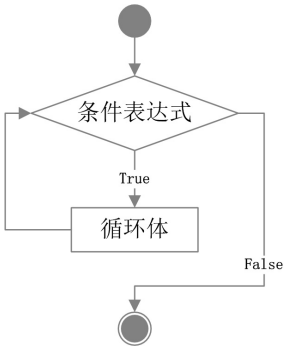


图 4.6　while 循环语句的执行流程图

我们以现实生活中的例子来理解 while 循环的执行流程。在体育课上，体育老师要求同学们沿着环形操场跑圈。要求当听到老师吹的哨子声时就停下来。同学们每跑一圈，可能会请求一次老师吹哨子。如果老师吹哨子，则停下来，即循环结束。否则继续跑步，即执行循环。

4.4.2　for循环

for 循环是一个计次循环，一般应用在循环次数已知的情况下。通常适用于枚举或遍历序列，以及迭代对象中的元素。语法如下：

```
for 迭代变量 in 对象:
    循环体
```

其中，迭代变量用于保存读取出的值；对象为要遍历或迭代的对象，该对象可以是任何有序的序列对象，如字符串、列表和元组等；循环体为一组被重复执行的语句。

for 循环语句的执行流程如图 4.7 所示。

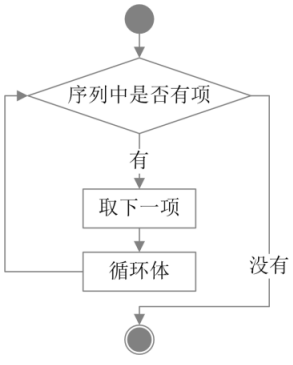


图 4.7　for 循环语句的执行流程图

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

我们以现实生活中的例子来理解 for 循环的执行流程。在体育课上，体育老师要求同学们排队进行踢毽球测试，每个同学一次机会，毽球落地则换另一个同学，直到全部同学都测试完毕，即循环结束。

1．进行数值循环

在使用 for 循环时，最基本的应用就是进行数值循环。例如，想要实现从 1 到 100 的累加，可以通过下面的代码实现。

```
01 print("计算 1+2+3+……+100 的结果为：")
02 result = 0          # 保存累加结果的变量
03 for i in range(101):
04     result += i     # 实现累加功能
05 print(result)       # 在循环结束时输出结果
```

在上面的代码中，使用了 range()函数，该函数是 Python 内置的函数，用于生成一系列连续的整数，多用于 for 循环语句中。其语法格式如下：

```
range(start,end,step)
```

各参数说明如下。

- start：用于指定计数的起始值，可以省略，如果省略则从 0 开始。
- end：用于指定计数的结束值（但不包括该值，如 range(7)，则得到的值为 0~6，不包括 7），不能省略。当 range()函数中只有一个参数时，即表示指定计数的结束值。
- step：用于指定步长，即两个数之间的间隔，可以省略，如果省略则表示步长为 1。例如，rang(1,7)将得到 1、2、3、4、5、6。

> **注意：**在使用 range()函数时，如果只有一个参数，那么表示指定的是 end；如果是两个参数，则表示指定的是 start 和 end；只有 3 个参数都存在时，最后一个才表示步长。

例如，使用下面的 for 循环语句，将输出 10 以内的所有奇数：

```
01 for i in range(1,10,2):
02     print(i,end = ' ')
```

得到的结果如下：

```
1 3 5 7 9
```

多学两招

在 Python 2.x 中，如果想让 print 语句输出的内容在一行上显示，可以在后面加上逗号（例如，print i,），但是在 Python 3.x 中，使用 print()函数时，不能直接加逗号，需要加上“,end = '分隔符'”，在上面的代码中使用的分隔符为一个空格。

> **说明：**在 Python 2.x 中，除提供 range()函数外，还提供了一个 xrange()函数，用于解决 range()函数会不经意间耗掉所有可用内存的问题，而在 Python 3.x 中已经更名为 range()函数，并且删除了老式 xrange()函数。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2．遍历字符串

使用 for 循环语句除了可以循环数值，还可以逐个遍历字符串，例如，下面的代码可以将横向显示的字符串转换为纵向显示：

```
01 string = '不要再说我不能'
02 print(string)          # 横向显示
03 for ch in string:
04     print(ch)          # 纵向显示
```

上面代码的运行结果如图 4.8 所示。

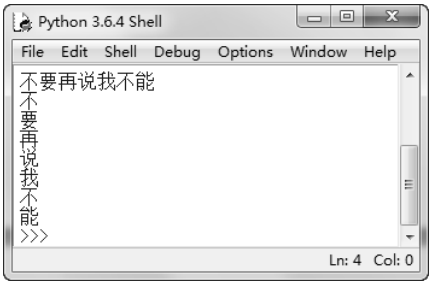


图 4.8　将字符串转换为纵向显示

说明：for 循环语句还可以用于迭代（遍历）列表、元组、集合和字典等。

4.4.3　循环嵌套

在 Python 中，允许在一个循环体中嵌入另一个循环，这称为循环嵌套。例如，在电影院找座位号，需要知道第几排第几列才能准确地找到自己的座位号，假如寻找如图 4.9 所示的在第二排第三列的座位号，首先寻找第二排，然后在第二排寻找第三列，这个寻找座位的过程就类似循环嵌套。

图 4.9　寻找座位的过程就类似循环嵌套

在 Python 中，for 循环和 while 循环都可以进行循环嵌套。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

例如，在 while 循环中套用 while 循环的格式如下：

```
while 条件表达式1:
    while 条件表达式2:
        循环体2
    循环体1
```

在 for 循环中套用 for 循环的格式如下：

```
for 迭代变量1 in 对象1:
    for 迭代变量2 in 对象2:
        循环体2
    循环体1
```

在 while 循环中套用 for 循环的格式如下：

```
while 条件表达式:
    for 迭代变量 in 对象:
        循环体2
    循环体1
```

在 for 循环中套用 while 循环的格式如下：

```
for 迭代变量 in 对象:
    while 条件表达式:
        循环体2
    循环体1
```

除了上面介绍的 4 种嵌套格式外，还可以实现更多层的嵌套，方法与上面的类似，这里就不再赘述了。

4.5 跳转语句

当循环条件一直满足时，程序将会一直执行下去，就像一辆迷路的车，在某个地方不停地转圆圈。如果希望在中间离开循环，也就是 for 循环结束计数之前，或者 while 循环找到结束条件之前。有两种方法能做到：

- 使用 continue 语句直接跳到循环的下一次迭代。
- 使用 break 语句完全中止循环。

下面将对 break 和 continue 语句进行详细介绍。

4.5.1 break语句

break 语句可以中止当前的循环，包括 while 和 for 在内的所有控制语句。以独自一人沿着操场跑步为例，原计划跑 10 圈，可是在跑到第 2 圈的时候，遇到自己的女神或者男神，于是果断停下来，中止跑步，这就相当于使用 break 语句提前中止了循环。break 语句的语法比较简单，只需要在相应的 while 或 for 语句中加入即可。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **说明：**break 语句一般会结合 if 语句进行搭配使用，表示在某种条件下，跳出循环。如果使用嵌套循环，break 语句将跳出最内层的循环。

在 while 语句中使用 break 语句的形式如下：

```
while 条件表达式 1:
    执行代码
    if 条件表达式 2:
        break
```

其中，条件表达式 2 用于判断何时调用 break 语句跳出循环。在 while 语句中使用 break 语句的流程如图 4.10 所示。

在 for 语句中使用 break 语句的形式如下：

```
for 迭代变量 in 对象:
    if 条件表达式:
        break
```

其中，条件表达式用于判断何时调用 break 语句跳出循环。在 for 语句中使用 break 语句的流程如图 4.11 所示。

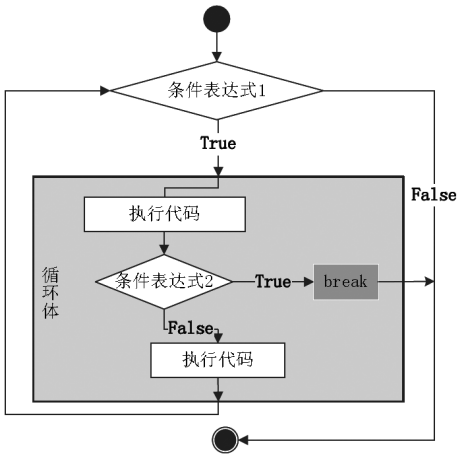


图 4.10　在 while 语句中使用 break 语句的流程图

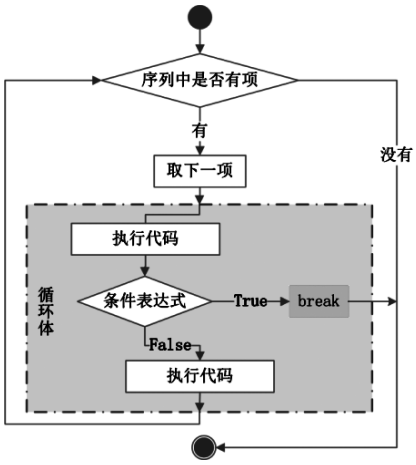


图 4.11　在 for 语句中使用 break 语句的流程图

4.5.2　continue语句

continue 语句的作用没有 break 语句强大，它只能中止本次循环而提前进入下一次循环中。仍然以独自一人沿着操场跑步为例，原计划跑步 10 圈，当跑到第 2 圈一半的时候，遇到自己的“女神”或者“男神”也在跑步，于是果断停下来，跑回起点等待，制造一次完美邂逅，然后从第 3 圈开始继续。

continue 语句的语法比较简单，只需要在相应的 while 或 for 语句中加入即可。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **说明：**continue 语句一般会结合 if 语句进行搭配使用，表示在某种条件下，跳过当前循环的剩余语句，然后继续进行下一轮循环。如果使用嵌套循环，continue 语句将只跳过最内层循环中的剩余语句。

在 while 语句中使用 continue 语句的形式如下：

```
while 条件表达式 1:
    执行代码
    if 条件表达式 2:
        continue
```

其中，条件表达式 2 用于判断何时调用 continue 语句跳出循环。在 while 语句中使用 continue 语句的流程如图 4.12 所示。

在 for 语句中使用 continue 语句的形式如下：

```
for 迭代变量 in 对象:
    if 条件表达式:
        continue
```

其中，条件表达式用于判断何时调用 continue 语句跳出循环。在 for 语句中使用 continue 语句的流程如图 4.13 所示。

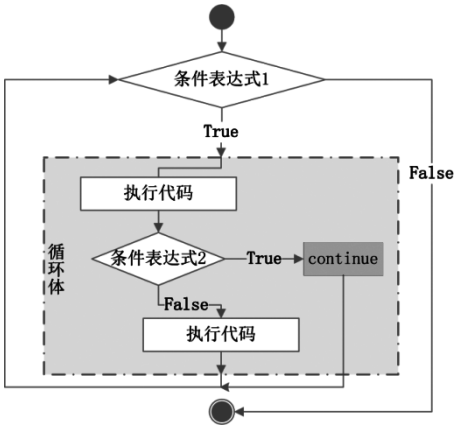


图 4.12　在 while 语句中使用 continue 语句的流程图

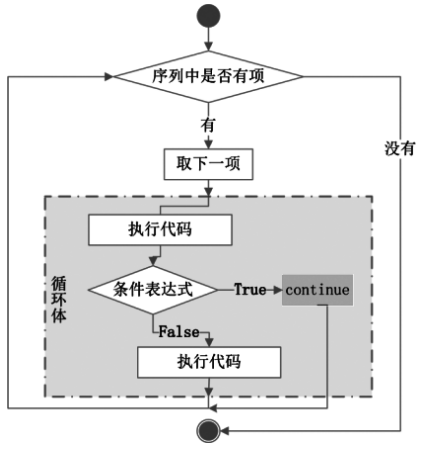


图 4.13　在 for 语句中使用 continue 语句的流程图

4.6　pass空语句

在 Python 中还有一个 pass 语句，表示空语句，它不做任何事情，一般起到占位作用。例如，在应用 for 循环输出 1～10 之间（不包括 10）的偶数时，在不是偶数时，应用 pass 语句占个位置，方便以后对不是偶数的数进行处理。代码如下：

```
01 for i in range(1,10):
02     if i%2 == 0:            # 判断是否为偶数
03         print(i,end = ' ')
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
04      else:                    # 不是偶数
05          pass                 # 占位符，不做任何事情
```

程序运行结果如下：

```
2 4 6 8
```

4.7　快学快用

（1）2017 年，全国零售百强中的 7 家电商保持较快增长，且增速相对于 2016 年显著提高，如图 4.14 所示。表 4.1 提供了 7 家企业的销售额和占比数据，请编写一个 Python 程序，将销售额超过 300 亿元的企业输出。

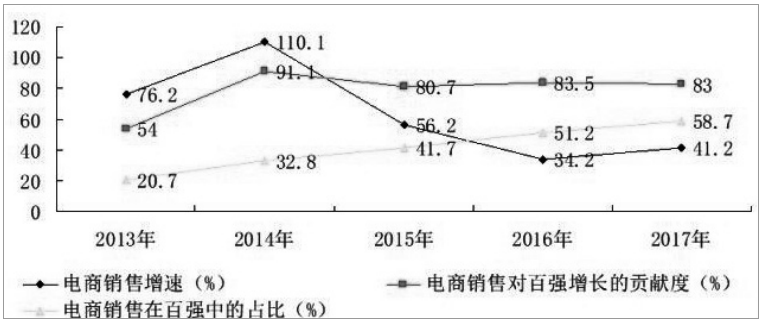


图 4.14　2013—2017 年电商销售增速、对百强增长的贡献度及占比

表 4.1　2017 年零售百强电商中销售额排名靠前的 7 家企业

企　业	聚美优品	京东	亚马逊中国	当当网	唯品会	考拉严选	天猫
销售额（亿元）	58	12945	391	357	728	116	21086
占比	1.9%	36.3%	1.1%	1.0%	0.3%	0.3%	59.1%

（2）计算今天是星期几。编写一个 Python 程序，输入数字 1～7 中的任意一个数（代表星期几），然后通过程序判断输出星期几。参考代码如下：

```
a = input("请输入数字（1——7）")
if int(a) == '1':
   print("今天是：星期一")
elif a == '2':
      print("今天是：星期二")
elif a == '3':
      print("今天是：星期三")
elif a == '4':
      print("今天是：星期四")
elif a == '5':
      print("今天是：星期五")
elif a == '6':
      print("今天是：星期六")
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
elif a == '7':
      print("今天是：星期日")
```

（3）Python 提供了很多有用的内置函数或模块，大家熟悉一下这些函数或模块的用途，在实际开发时如果遇到相关应用，可以联想到相关函数或模块，然后详细学习并使用，事半功倍。数学运算类内置函数如表 4.2 所示。如果要详细了解这些函数或模块的使用，可以扫描二维码，到快用云端支持平台学习、训练、实战。

表 4.2　数学运算类内置函数（3）

pow(x, y[, z])	返回 x 的 y 次幂
range([start], stop[, step])	产生一个序列，默认从 0 开始
round(x[, n])	四舍五入
sum(iterable[, start])	对集合求和
bin(x)	将整数 x 转换为二进制字符串
bool([x])	将 x 转换为布尔类型

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 2 篇　进 阶 篇

第 5 章

序列的应用

在数学里，序列也称为数列，是指按照一定顺序排列的一列数，而在程序设计中，序列是一种常用的数据存储方式，几乎每一种程序设计语言都提供了类似的数据结构。例如，C 语言或 Java 中的数组等。

在 Python 中序列是最基本的数据结构，它是一块用于存放多个值的连续内存空间。Python 中内置了 5 个常用的序列结构，分别是列表、元组、字典和字符串等。

5.1　序列

序列是一块用于存放多个值的连续内存空间，并且按一定顺序排列，每一个值（称为元素）都分配一个数字，称为索引或位置，通过该索引可以取出相应的值。例如，我们可以把一家酒店看作一个序列，那么酒店里的每个房间都可以看作是这个序列的元素，而房间号就相当于索引，可以通过房间号找到对应的房间。

在 Python 中，序列结构主要有列表、元组、集合和字典。对于这些序列结构有以下几个通用的操作。

5.1.1　索引（Indexing）

序列中的每一个元素都有一个编号，也称为索引。这个索引是从 0 开始递增的，即下标为 0 表示第一个元素，下标为 1 表示第 2 个元素，依次类推，如图 5.1 所示。

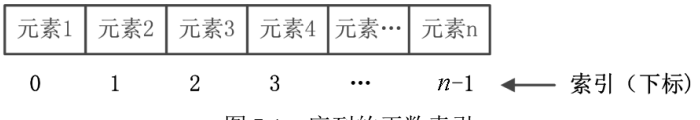


图 5.1　序列的正数索引

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

Python 比较神奇，它的索引可以是负数，这个索引从右向左计数，也就是从最后一个元素开始计数，即最后一个元素的索引值为-1，倒数第二个元素的索引值为-2，依次类推，如图 5.2 所示。

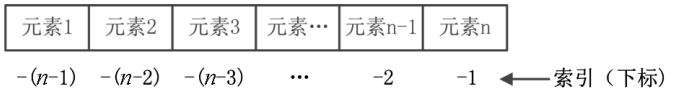


图 5.2　序列的负数索引

> **注意**：在采用负数作为索引值时，是从-1 开始的，而不是从 0 开始的，即最后一个元素的下标为-1，这是为了防止与第一个元素重合。

通过索引可以访问序列中的任何元素。例如，定义一个包括 4 个元素的列表，要访问它的第 3 个元素和最后一个元素，可以使用下面的代码：

```
verse = ["春眠不觉晓","Python 不得了","夜来爬数据","好评知多少"]
print(verse[2])     # 输出第 3 个元素
print(verse[-1])    # 输出最后一个元素
```

输出的结果为显示文字：

```
夜来爬数据
好评知多少
```

> **说明**：关于列表的详细介绍请参见 5.2 节。

5.1.2　切片（Slicing）

切片操作是访问序列中元素的另一种方法，它可以访问一定范围内的元素。通过切片操作可以生成一个新的序列。实现切片操作的语法格式如下：

```
sname[start : end : step]
```

参数说明如下。

- sname：表示序列的名称。
- start：表示切片的开始位置（包括该位置），如果不指定，则默认为 0。
- end：表示切片的截止位置（不包括该位置），如果不指定则默认为序列的长度。
- step：表示切片的步长，如果省略，则默认为 1，当省略该步长时，最后一个冒号也可以省略。

> **说明**：在进行切片操作时，如果指定了步长，那么将按照该步长遍历序列的元素，否则将一个一个遍历序列。

例如，通过切片获取 NBA 历史上十大巨星列表中的第 2 个到第 5 个元素，以及获取第 1 个、第 3 个和第 5 个元素，可以使用下面的代码：

```
nba = ["迈克尔·乔丹","比尔·拉塞尔","卡里姆阿卜杜勒·贾巴尔","威尔特·张伯伦",
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
02       "埃尔文·约翰逊","科比·布莱恩特","蒂姆·邓肯","勒布朗·詹姆斯","拉里·伯德",
03       "沙奎尔·奥尼尔"]
04  print(nba[1:5])          # 获取第 2 个到第 5 个元素
05  print(nba[0:2:5])        # 获取第 1 个、第 3 个和第 5 个元素
```

运行上面的代码，将输出以下内容：

```
['比尔·拉塞尔', '卡里姆阿卜杜勒·贾巴尔', '威尔特·张伯伦', '埃尔文·约翰逊']
['迈克尔·乔丹', '卡里姆阿卜杜勒·贾巴尔', '埃尔文·约翰逊']
```

> **说明：**如果想要复制整个序列，可以将 start 和 end 参数都省略，但是中间的冒号需要保留。例如，verse[:]就表示复制整个名称为 verse 的序列。

5.1.3　序列相加（Adding）

在 Python 中，支持两种相同类型的序列相加操作，即将两个序列进行连接，使用加（+）运算符实现。例如，将两个列表相加，可以使用下面的代码：

```
01  nba1 = ["德怀特·霍华德","德维恩·韦德","凯里·欧文","保罗·加索尔"]
02  nba2 = ["迈克尔·乔丹","比尔·拉塞尔","卡里姆阿卜杜勒·贾巴尔","威尔特·张伯伦",
03       "埃尔文·约翰逊","科比·布莱恩特","蒂姆·邓肯","勒布朗·詹姆斯","拉里·伯德",
04       "沙奎尔·奥尼尔"]
05  print(nba1+nba2)
```

运行上面的代码，将输出以下内容：

```
['德怀特·霍华德', '德维恩·韦德', '凯里·欧文', '保罗·加索尔', '迈克尔·乔丹', '比尔·拉塞尔', '卡里姆阿卜杜勒·贾巴尔', '威尔特·张伯伦', '埃尔文·约翰逊', '科比·布莱恩特', '蒂姆·邓肯', '勒布朗·詹姆斯', '拉里·伯德', '沙奎尔·奥尼尔']
```

从上面的输出结果中可以看出，两个列表被合为一个列表了。

> **说明：**在进行序列相加时，相同类型的序列是指同为列表、元组、集合等，序列中的元素类型可以不同。例如，下面的代码也是正确的：
>
> ```
> 01    num = [7,14,21,28,35,42,49,56]
> 02    nba = ["德怀特·霍华德","德维恩·韦德","凯里·欧文","保罗·加索尔"]
> 03    print(num + nba)
> ```
>
> 相加后的结果如下：
>
> ```
> [7, 14, 21, 28, 35, 42, 49, 56, '德怀特·霍华德', '德维恩·韦德', '凯里·欧文', '保罗·加索尔']
> ```
>
> 但是不能是列表和元组相加，或者列表和字符串相加。例如，下面的代码就是错误的：
>
> ```
> 01    num = [7,14,21,28,35,42,49,56,63]
> 02    print(num + "输出是 7 的倍数的数")
> ```
>
> 上面的代码，在运行后，将产生如图 5.3 所示的异常信息。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
Traceback (most recent call last):
  File "E:\program\Python\Code\datatype_test.py", line 2, in <module>
    print(num + "输出是7的倍数的数")
TypeError: can only concatenate list (not "str") to list
>>>
```

图 5.3　将列表和字符串相加产生的异常信息

5.1.4　乘法（Multiplying）

在 Python 中，使用数字 *n* 乘以一个序列会生成新的序列，新序列的内容为原来序列被重复 *n* 次的结果。例如，下面的代码，将实现将一个序列乘以 3 生成一个新的序列并输出，从而达到重要事情说三遍的效果。

```
01  phone = ["华为 Mate 10","Vivo X21"]
02  print(phone * 3)
```

运行上面的代码，将显示以下内容：

```
['华为 Mate 10', 'Vivo X21', '华为 Mate 10', 'Vivo X21', '华为 Mate 10', 'Vivo X21']
```

在进行序列的乘法运算时，还可以实现初始化指定长度列表的功能。例如下面的代码，将创建一个长度为 5 的列表，列表的每个元素都是 None，表示什么都没有。

```
01  emptylist = [None]*5
02  print(emptylist)
```

运行上面的代码，将显示以下内容：

```
[None, None, None, None, None]
```

5.1.5　检查某个元素是否是序列的成员（元素）

在 Python 中，可以使用 in 关键字检查某个元素是否是序列的成员，即检查某个元素是否包含在该序列中。语法格式如下：

```
value in sequence
```

其中，value 表示要检查的元素；sequence 表示指定的序列。

例如，要检查名称为 nba 的序列中是否包含元素“保罗·加索尔”，可以使用下面的代码：

```
01  nba = ["德怀特·霍华德","德维恩·韦德","凯里·欧文","保罗·加索尔"]
02  print("保罗·加索尔" in nba)
```

运行上面的代码，将显示 True，表示在序列中存在指定的元素。

另外，在 Python 中，也可以使用 not in 关键字实现检查某个元素是否不包含在指定的序列中。例如下面的代码，将显示 False：

```
01  nba = ["德怀特·霍华德","德维恩·韦德","凯里·欧文","保罗·加索尔"]
02  print(保罗·加索尔"  not in nba)
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

5.1.6　计算序列的长度、最大值和最小值

在 Python 中，提供了内置函数计算序列的长度、最大值和最小值，分别是使用 len()函数计算序列的长度，即返回序列包含多少个元素；使用 max()函数返回序列中的最大元素；使用 min()函数返回序列中的最小元素。

例如，定义一个包括 9 个元素的列表，并通过 len()函数计算列表的长度，可以使用下面的代码：

```
01 num = [7,14,21,28,35,42,49,56,63]
02 print("序列 num 的长度为",len(num))
```

运行上面的代码，将显示“序列 num 的长度为 9”。

例如，定义一个包括 9 个元素的列表，并通过 max()函数计算列表中的最大元素，可以使用下面的代码：

```
01 num = [7,14,21,28,35,42,49,56,63]
02 print("序列",num,"中最大值为",max(num))
```

运行上面的代码，将显示以下结果：

```
序列 [7, 14, 21, 28, 35, 42, 49, 56, 63] 中最大值为 63
```

例如，定义一个包括 9 个元素的列表，并通过 min()函数计算列表的最小元素，可以使用下面的代码：

```
01 num = [7,14,21,28,35,42,49,56,63]
02 print("序列",num,"中最小值为",min(num))
```

运行上面的代码，将显示以下结果：

```
序列 [7, 14, 21, 28, 35, 42, 49, 56, 63] 中最小值为 7
```

除了上面介绍的 3 个内置函数，Python 还提供了如表 5.1 所示的内置函数。

表 5.1　Python 提供的内置函数及其作用

函　数	作　用
list()	将序列转换为列表
str()	将序列转换为字符串
sum()	计算元素和
sorted()	对元素进行排序
reversed()	反向序列中的元素
enumerate()	将序列组合为一个索引序列，多用在 for 循环中

5.2　列表（list）

对于歌曲列表大家一定很熟悉，在列表中记录着要播放的歌曲名称，如图 5.4 所示为手机 App 中的歌曲列表页面。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 5.4　歌曲列表

在 Python 中的列表和歌曲列表类似，也是由一系列按特定顺序排列的元素组成，它是 Python 中内置的可变序列。在形式上，列表的所有元素都放在一对中括号“[]”中，两个相邻元素间使用逗号“,”进行分隔。在内容上，可以将整数、实数、字符串、列表、元组等任何类型的内容放入列表中，并且同一个列表中，元素的类型可以不同，因为它们之间没有任何关系。由此可见，Python 中的列表是非常灵活的，这一点与其他语言是不同的。

5.2.1　列表的创建和删除

在 Python 中提供了多种创建列表的方法，下面分别进行介绍。

1．使用赋值运算符直接创建列表

同其他类型的 Python 变量一样，创建列表时，也可以使用赋值运算符“=”直接将一个列表赋值给变量。具体的语法格式如下：

```
listname = [element 1,element 2,element 3,…,element n]
```

其中，listname 表示列表的名称，可以是任何符合 Python 命名规则的标识符；element 1、element 2、element 3、element n 表示列表中的元素，个数没有限制，并且只要是 Python 支持的数据类型就可以。

例如，下面定义的都是合法的列表：

```
01  num = [7,14,21,28,35,42,49,56,63]
02  verse = ["自古逢秋悲寂寥","我言秋日胜春朝","晴空一鹤排云上","便引诗情到碧宵"]
03  untitle = ['Python',28,"人生苦短，我用 Python",["爬虫","自动化运维","云计算","Web 开发"]]
04  python = ['优雅',"明确",'''简单''']
```

> **说明：**在使用列表时，虽然可以将不同类型的数据放入到同一个列表中，但是通常情况下，我们不这样做，而是在一个列表中只放入一种类型的数据，这样可以提高程序的可读性。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2．创建空列表

在 Python 中，也可以创建空列表，例如，要创建一个名称为 emptylist 的空列表，可以使用下面的代码：

```
emptylist = []
```

3．创建数值列表

在 Python 中，数值列表很常用。例如，在考试系统中记录学生的成绩，或者在游戏中记录每个角色的位置、各个玩家的得分情况等。在 Python 中，可以使用 list()函数直接将 range()函数循环出来的结果转换为列表。

list()函数的基本语法如下：

```
list(data)
```

其中，data 表示可以转换为列表的数据，其类型可以是 range 对象、字符串、元组或者其他可迭代类型的数据。

例如，创建一个 10～20 之间（不包括 20）所有偶数的列表，可以使用下面的代码：

```
list(range(10, 20, 2))
```

运行上面的代码，将得到下面的列表：

```
[10, 12, 14, 16, 18]
```

> **说明：**使用 list()函数不仅能通过 range 对象创建列表，还可以通过其他对象创建列表。

4．删除列表

对于已经创建的列表，不再使用时，可以使用 del 语句将其删除，语法格式如下：

```
del listname
```

其中，listname 为要删除列表的名称。

> **说明：**del 语句在实际开发时并不常用，因为 Python 自带的垃圾回收机制会自动销毁不用的列表，所以即使我们不手动将其删除，Python 也会自动将其回收。

例如，定义一个名称为 team 的列表，然后再用 del 语句将其删除，可以使用下面的代码：

```
01 team = ["皇马","罗马","利物浦","拜仁"]
02 del team
```

常见错误

在删除列表前，一定要保证输入的列表名称是已经存在的，否则将出现如图 5.5 所示的错误。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

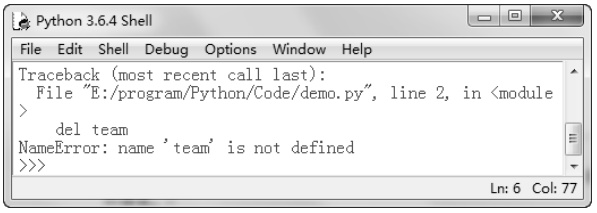


图 5.5　删除的列表不存在产生的异常信息

5.2.2　访问列表元素

在 Python 中，如果想将列表的内容输出也比较简单，可以直接使用 print()函数。例如，想打印上面列表中的 untitle 列表，则可以使用下面的代码：

```
print(untitle)
```

执行结果如下：

```
['Python', 28, '人生苦短，我用 Python', ['爬虫', '自动化运维', '云计算', 'Web 开发', '游戏']]
```

从上面的执行结果中可以看出，在输出列表时，是包括左右两侧的中括号的。如果不想输出全部的元素，也可以通过列表的索引获取指定的元素。例如，要获取列表 untitle 中索引为 2 的元素，可以使用下面的代码：

```
print(untitle[2])
```

执行结果如下：

```
人生苦短，我用 Python
```

从上面的执行结果中可以看出，在输出单个列表元素时，不包括中括号。如果是字符串，还不包括左右的引号。

5.2.3　遍历列表

遍历列表中的所有元素是常用的一种操作，在遍历的过程中可以完成查询、处理等功能。在生活中，如果想要去商场买一件衣服，就需要在商场中逛一遍，看是否有想要的衣服，逛商场的过程就相当于列表的遍历操作。在 Python 中遍历列表的方法有多种，下面介绍两种常用的方法。

1．直接使用 for 循环实现

直接使用 for 循环遍历列表，只能输出元素的值，它的语法格式如下：

```
for item in listname:
    # 输出 item
```

其中，item 用于保存获取到的元素值，要输出元素内容时，直接输出该变量即可；listname 为列表名称。

例如，定义一个保存 2018 年 NBA 常规赛西部球队排名的列表，然后通过 for 循环遍历该

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

列表，并输出各个球队名称，代码如下：

```
01 print("2018 年 NBA 常规赛西部排名：")
02 team = ["休斯敦 火箭","金州 勇士","波特兰 开拓者","犹他 爵士","新奥尔良 鹈鹕","圣安东尼奥 马刺","俄克拉荷马城 雷霆","明尼苏达 森林狼"]
03 for item in team:
04     print(item)
```

执行上面的代码，将显示如图 5.6 所示的结果。

图 5.6　通过 for 循环遍历列表

2．使用 for 循环和 enumerate()函数实现

使用 for 循环和 enumerate()函数可以实现同时输出索引值和元素内容，它的语法格式如下：

```
# 输出 index 和 item
for index,item in enumerate(listname):
```

参数说明如下。

- index：用于保存元素的索引。
- item：用于保存获取到的元素值，要输出元素内容时，直接输出该变量即可。
- listname：为列表名称。

例如，定义一个保存 2018 年 NBA 常规赛西部球队排名的列表，然后通过 for 循环和 enumerate()函数遍历该列表，并输出索引和球队名称，代码如下：

```
01 print("2018 年 NBA 常规赛西部排名：")
02 team = ["休斯敦 火箭","金州 勇士","波特兰 开拓者","犹他 爵士","新奥尔良 鹈鹕","圣安东尼奥 马刺","俄克拉荷马城 雷霆","明尼苏达 森林狼"]
03 for index,item in enumerate(team):
04     print(index + 1,item)
```

执行上面的代码，将显示下面的结果：

```
2018 年 NBA 常规赛西部排名：
1 休斯敦 火箭
2 金州 勇士
3 波特兰 开拓者
4 犹他 爵士
5 新奥尔良 鹈鹕
6 圣安东尼奥 马刺
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
7 俄克拉荷马城 雷霆
8 明尼苏达 森林狼
```

5.2.4 添加、修改和删除列表元素

添加、修改和删除列表元素也称为更新列表。在实际开发时，经常需要对列表进行更新。下面我们就分别介绍如何实现列表元素的添加、修改和删除。

1. 添加元素

在 5.1.3 节介绍了通过“+”号将两个序列连接，该方法也可以实现为列表添加元素。但是这种方法的执行速度要比直接使用列表对象的 append()方法慢，所以建议在实现添加元素时，使用列表对象的 append()方法实现。列表对象的 append()方法用于在列表的末尾追加元素，它的语法格式如下：

```
listname.append(obj)
```

其中，listname 为要添加元素的列表名称；obj 为要添加到列表末尾的对象。

例如，定义一个包括 4 个元素的列表，然后应用 append()方法向该列表的末尾再添加一个元素，可以使用下面的代码：

```
01 phone = ["摩托罗拉","诺基亚","三星","OPPO"]
02 len(phone)       # 获取列表的长度
03 phone.append("iPhone")
04 len(phone)       # 获取列表的长度
05 print(phone)
```

上面的代码在 IDEL 的 Shell 窗口中一行一行执行的过程如图 5.7 所示。

图 5.7 向列表中添加元素

多学两招

列表对象除了提供 append()方法可以向列表中添加元素外，还提供了 insert()方法也可以向列表中添加元素，该方法用于向列表的指定位置插入元素。但是由于该方法的执行效率没有 append()方法高，所以不推荐这种方法。

上面介绍的是向列表中添加元素，如果想要将一个列表中的全部元素添加到另一个列表中，可以使用列表对象的 extend()方法实现。extend()方法的具体语法如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
listname.extend(seq)
```

其中，listname 为原列表；seq 为要添加的列表。语句执行后，seq 的内容将追加到 listname 的后面。

2．修改元素

修改列表中的元素只需要通过索引获取该元素，然后再为其重新赋值即可。例如，定义一个保存 3 个元素的列表，然后修改索引值为 2 的元素，代码如下：

```
01 verse = ["长亭外","古道边","芳草碧连天"]
02 print(verse)
03 verse[2] = "一行白鹭上青天"    # 修改列表的第 3 个元素
04 print(verse)
```

上面的代码在 IDEL 中的执行过程如图 5.8 所示。

```
>>> verse = ["长亭外","古道边","芳草碧连天"]
>>> print(verse)
['长亭外', '古道边', '芳草碧连天']
>>> verse[2] = "一行白鹭上青天"
>>> print(verse)
['长亭外', '古道边', '一行白鹭上青天']
>>>
```

图 5.8　修改列表的指定元素

3．删除元素

删除元素主要有两种情况，一种是根据索引删除，另一种是根据元素值删除。下面分别进行介绍。

（1）根据索引删除

删除列表中的指定元素和删除列表类似，也可以使用 del 语句实现，所不同的就是在指定列表名称时，换为列表元素。例如，定义一个保存 3 个元素的列表，删除最后一个元素，可以使用下面的代码：

```
01 verse = ["长亭外","古道边","芳草碧连天"]
02 del verse[-1]
03 print(verse)
```

上面的代码在 IDEL 中的执行过程如图 5.9 所示。

```
>>> verse = ["长亭外","古道边","芳草碧连天"]
>>> del verse[-1]
>>> print(verse)
['长亭外', '古道边']
>>>
```

图 5.9　删除列表的指定元素

（2）根据元素值删除

如果想要删除一个不确定其位置的元素（根据元素值删除），可以使用列表对象的 remove() 方法实现。例如，要删除列表中内容为“古道边”的元素，可以使用下面的代码：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
verse = ["长亭外","古道边","芳草碧连天"]
verse.remove("古道边")
```

使用列表对象的 remove()方法删除元素时，如果指定的元素不存在，将出现如图 5.10 所示的异常信息。

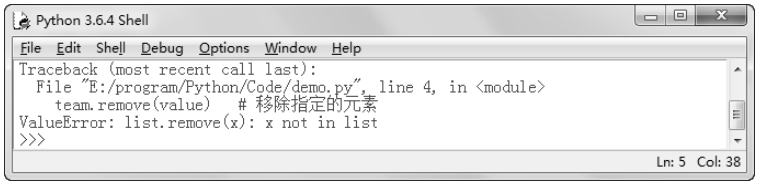


图 5.10　删除不存在的元素时出现的异常信息

所以在使用 remove()方法删除元素前，最好先判断该元素是否存在，例如下面的代码：

```
team = ["火箭","勇士","开拓者","爵士","鹈鹕","马刺","雷霆","森林狼"]
value = "公牛"                  # 指定要移除的元素
if team.count(value)>0:         # 判断要删除的元素是否存在
    team.remove(value)          # 移除指定的元素
print(team)
```

> **说明：**列表对象的 count()方法用于判断指定元素出现的次数，返回结果为 0 时，表示不存在该元素。关于 count()方法的详细介绍请参见 5.2.5 节。

执行上面的代码后，将显示下面的列表原有内容：

```
['火箭', '勇士', '开拓者', '爵士', '鹈鹕', '马刺', '雷霆', '森林狼']
```

5.2.5　对列表进行统计计算

Python 的列表提供了内置的一些函数来实现统计方面的功能。下面介绍常用的功能。

1．获取指定元素出现的次数

使用列表对象的 count()方法可以获取指定元素在列表中出现的次数，基本语法格式如下：

```
listname.count(obj)
```

其中，listname 表示列表的名称；obj 表示要判断是否存在的对象，这里只能进行精确匹配，即不能是元素值的一部分。

例如，创建一个列表，内容为听众点播的歌曲列表，然后应用列表对象的 count()方法判断元素“云在飞”出现的次数，代码如下：

```
song = ["云在飞","我在诛仙逍遥涧","送你一匹马","半壶纱","云在飞","遇见你","等你等了那么久"]
num = song.count("云在飞")
print(num)
```

上面的代码运行后，将显示 2，表示在列表 song 中“云在飞”出现了两次。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2．获取指定元素首次出现的下标

使用列表对象的 index()方法可以获取指定元素在列表中首次出现的位置（索引），基本语法格式如下：

```
listname.index(obj)
```

参数说明如下。

- listname：表示列表的名称。
- obj：表示要查找的对象，这里只能进行精确匹配。如果指定的对象不存在，则抛出如图 5.11 所示的异常。
- 返回值：首次出现的索引值。

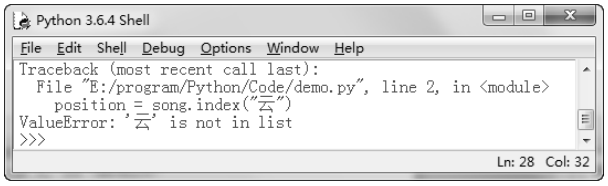


图 5.11　查找对象不存在时抛出的异常

例如，创建一个列表，内容为听众点播的歌曲列表，然后应用列表对象的 index()方法判断元素“半壶纱”首次出现的位置，代码如下：

```
01 song = ["云在飞","我在诛仙逍遥涧","送你一匹马","半壶纱","云在飞","遇见你","等你等了那么久"]
02 position = song.index("半壶纱")
03 print(position)
```

上面的代码运行后，将显示 3，表示“半壶纱”在列表 song 中首次出现的索引位置是 3。

3．统计数值列表的元素和

在 Python 中，提供了 sum()函数用于统计数值列表中各元素的和，语法格式如下：

```
sum(iterable[,start])
```

参数说明如下。

- iterable：表示要统计的列表。
- start：表示统计结果是从哪个数开始（将统计结果加上 start 所指定的数），是可选参数，如果没有指定，默认值为 0。

例如，定义一个保存 10 名学生语文成绩的列表，然后应用 sum()函数统计列表中元素的和，即统计总成绩，然后输出，代码如下：

```
01 grade = [98,99,97,100,100,96,94,89,95,100]      # 10 名学生语文成绩列表
02 total = sum(grade)                              # 计算总成绩
03 print("语文总成绩为：",total)
```

上面的代码执行后，将显示“语文总成绩为： 968”。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

5.2.6 对列表进行排序

在实际开发时，经常需要对列表进行排序。Python 中提供了两种常用的对列表进行排序的方法。下面分别进行介绍。

1．使用列表对象的 sort()方法实现

列表对象提供了 sort()方法用于对原列表中的元素进行排序。排序后原列表中的元素顺序将发现改变。列表对象的 sort()方法的语法格式如下：

```
listname.sort(key=None, reverse=False)
```

参数说明如下。

- listname：表示要进行排序的列表。
- key：表示指定一个从每个列表元素中提取一个比较键（例如，设置“key=str.lower”表示在排序时不区分字母大小写）。
- reverse：可选参数，如果将其值指定为 True，则表示降序排列；如果将其值指定为 False，则表示升序排列。默认为升序排列。

例如，定义一个保存 10 名学生语文成绩的列表，然后应用 sort()方法对其进行排序，代码如下：

```
01 grade = [98,99,97,100,100,96,94,89,95,100]     # 10 名学生语文成绩列表
02 print("原列表：",grade)
03 grade.sort()                                    # 进行升序排列
04 print("升　序：",grade)
05 grade.sort(reverse=True)                        # 进行降序排列
06 print("降　序：",grade)
```

执行上面的代码，将显示以下内容：

```
原列表： [98, 99, 97, 100, 100, 96, 94, 89, 95, 100]
升　序： [89, 94, 95, 96, 97, 98, 99, 100, 100, 100]
降　序： [100, 100, 100, 99, 98, 97, 96, 95, 94, 89]
```

使用 sort()方法进行数值列表的排序比较简单，但是使用 sort()方法对字符串列表进行排序时，采用的规则是先对大写字母进行排序，然后再对小写字母进行排序。如果想要对字符串列表进行排序（不区分大小写时），需要指定其 key 参数。例如，定义一个保存英文字符串的列表，然后应用 sort()方法对其进行升序排列，可以使用下面的代码：

```
01 char = ['cat','Tom','Angela','pet']
01 char.sort()                      # 默认区分字母大小写
02 print("区分字母大小写：",char)
03 char.sort(key=str.lower)         # 不区分字母大小写
04 print("不区分字母大小写：",char)
```

运行上面的代码，将显示以下内容：

```
区分字母大小写： ['Angela', 'Tom', 'cat', 'pet']
不区分字母大小写： ['Angela', 'cat', 'pet', 'Tom']
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **说明：**采用 sort()方法对列表进行排序时，对于中文支持不好。排序的结果与我们常用的按拼音或者笔画都不一致。如果需要实现对中文内容的列表排序，还需要重新编写相应的方法进行处理，不能直接使用 sort()方法。

2．使用内置的 sorted()函数实现

在 Python 中，提供了一个内置的 sorted()函数，用于对列表进行排序。使用该函数进行排序后，原列表的元素顺序不变。sorted()函数的语法格式如下：

```
sorted(iterable, key=None, reverse=False)
```

参数说明如下。

- iterable：表示要进行排序的列表名称。
- key：表示指定一个从每个列表元素中提取一个比较键（例如，设置“key=str.lower”表示在排序时不区分字母大小写）。
- reverse：可选参数，如果将其值指定为 True，则表示降序排列；如果将其值指定为 False，则表示升序排列。默认为升序排列。

例如，定义一个保存 10 名学生语文成绩的列表，然后应用 sorted()函数对其进行排序，代码如下：

```
grade = [98,99,97,100,100,96,94,89,95,100]      # 10 名学生语文成绩列表
grade_as = sorted(grade)                        # 进行升序排列
print("升序：",grade_as)
grade_des = sorted(grade,reverse = True)        # 进行降序排列
print("降序：",grade_des)
print("原序列：",grade)
```

执行上面的代码，将显示以下内容：

```
升序： [89, 94, 95, 96, 97, 98, 99, 100, 100, 100]
降序： [100, 100, 100, 99, 98, 97, 96, 95, 94, 89]
原序列： [98, 99, 97, 100, 100, 96, 94, 89, 95, 100]
```

> **说明：**列表对象的 sort()方法和内置 sorted()函数的作用基本相同，所不同的就是使用 sort()方法时，会改变原列表的元素排列顺序，但是使用 storted()函数时，会建立一个原列表的副本，该副本为排序后的列表。

5.2.7　列表推导式

使用列表推导式可以快速生成一个列表，或者根据某个列表生成满足指定需求的列表。列表推导式通常有以下几种常用的语法格式。

（1）生成指定范围的数值列表，语法格式如下：

```
list = [Expression for var in range]
```

参数说明如下。

- list：表示生成的列表名称。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- Expression：表达式，用于计算新列表的元素。
- var：循环变量。
- range：采用 range()函数生成的 range 对象。

例如，要生成一个包括 10 个随机数的列表，要求数的范围在 10～100（包括）之间，具体代码如下：

```
01 import random                                          #导入 random 标准库
02 randomnumber = [random.randint(10,100) for i in range(10)]
03 print("生成的随机数为：",randomnumber)
```

执行结果如下：

```
生成的随机数为： [38, 12, 28, 26, 58, 67, 100, 41, 97, 15]
```

（2）根据列表生成指定需求的列表，语法格式如下：

```
newlist = [Expression for var in list]
```

参数说明如下。

- newlist：表示新生成的列表名称。
- Expression：表达式，用于计算新列表的元素。
- var：变量，值为后面列表的每个元素值。
- list：用于生成新列表的原列表。

例如，定义一个记录商品价格的列表，然后应用列表推导式生成一个将全部商品价格打五折的列表，具体代码如下：

```
01 price = [1200,5330,2988,6200,1998,8888]
02 sale = [int(x*0.5) for x in price]
03 print("原价格：",price)
04 print("打五折的价格：",sale)
```

执行结果如下：

```
原价格： [1200, 5330, 2988, 6200, 1998, 8888]
打五折的价格： [600, 2665, 1494, 3100, 999, 4444]
```

（3）从列表中选择符合条件的元素组成新的列表，语法格式如下：

```
newlist = [Expression for var in list if condition]
```

参数说明如下。

- newlist：表示新生成的列表名称。
- Expression：表达式，用于计算新列表的元素。
- var：变量，值为后面列表的每个元素值。
- list：用于生成新列表的原列表。
- condition：条件表达式，用于指定筛选条件。

例如，定义一个记录商品价格的列表，然后应用列表推导式生成一个商品价格高于 5000 的列表，具体代码如下：

```
01 price = [1200,5330,2988,6200,1998,8888]
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
02  sale = [x for x in price if x>5000]
03  print("原列表：",price)
04  print("价格高于 5000 的：",sale)
```

执行结果如下：

```
原列表： [1200, 5330, 2988, 6200, 1998, 8888]
价格高于 5000 的： [5330, 6200, 8888]
```

5.3　元组（tuple）

元组（tuple）是 Python 中另一个重要的序列结构，与列表类似，也是由一系列按特定顺序排列的元素组成的，但是它是不可变序列，因此，元组也可以称为不可变的列表。在形式上，元组的所有元素都放在一对小括号“()”中，两个相邻元素间使用逗号“,”进行分隔。在内容上，可以将整数、实数、字符串、列表、元组等任何类型的内容放入元组中，并且同一个元组中，元素的类型可以不同，因为它们之间没有任何关系。在通常情况下，元组用于保存程序中不可修改的内容。

> **说明：**从元组和列表的定义上看，这两种结构比较相似，那么它们之间有哪些区别呢？它们之间的主要区别就是一个是不可变序列，另一个是可变序列，即元组中的元素不可以单独修改，而列表则可以任意修改。

5.3.1　元组的创建和删除

在 Python 中提供了多种创建元组的方法，下面分别进行介绍。

1．使用赋值运算符直接创建元组

同其他类型的 Python 变量一样，创建元组时，也可以使用赋值运算符“=”直接将一个元组赋值给变量。具体的语法格式如下：

```
tuplename = (element 1,element 2,element 3,…,element n)
```

其中，tuplename 表示元组的名称，可以是任何符合 Python 命名规则的标识符；element 1、element 2、element 3、element n 表示元组中的元素，个数没有限制，并且只要是 Python 支持的数据类型就可以。

> **注意：**创建元组的语法与创建列表的语法类似，只是创建列表时使用的是中括号“[]”，而创建元组时使用的是小括号“()”。

例如，下面定义的都是合法的元组：

```
01  num = (7,14,21,28,35,42,49,56,63)
02  ukguzheng = ("渔舟唱晚","高山流水","出水莲","汉宫秋月")
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
03 untitle = ('Python',28,("人生苦短","我用 Python"),["爬虫","自动化运维","云计算","Web 开发"])
04 python = ('优雅',"明确",'''简单''')
```

在 Python 中，虽然元组是使用一对小括号将所有的元素括起来的，但是实际上，小括号并不是必需的，只要将一组值用逗号分隔开来，Python 就可以认为它是元组。例如，下面的代码定义的也是元组。

```
ukguzheng = "渔舟唱晚","高山流水","出水莲","汉宫秋月"
```

在 IDLE 中输出该元组后，将显示以下内容：

```
('渔舟唱晚', '高山流水', '出水莲', '汉宫秋月')
```

如果要创建的元组只包括一个元素，则需要在定义元组时，在元素的后面加一个逗号“,”。例如，下面的代码定义的就是包括一个元素的元组：

```
verse = ("一片冰心在玉壶",)
```

在 IDLE 中输出 verse，将显示以下内容：

```
('一片冰心在玉壶',)
```

而下面的代码，则表示定义一个字符串：

```
verse = ("一片冰心在玉壶")
```

在 IDLE 中输出 verse，将显示以下内容：

```
一片冰心在玉壶
```

> **说明**：在 Python 中，可以使用 type()函数测试变量的类型。例如，下面的代码：
>
> ```
> 01 verse1 = ("一片冰心在玉壶",)
> 02 print("verse1 的类型为",type(verse1))
> 03 verse2 = ("一片冰心在玉壶")
> 04 print("verse2 的类型为",type(verse2))
> ```

在 IDLE 中执行上面的代码，将显示以下内容：

```
verse1 的类型为 <class 'tuple'>
verse2 的类型为 <class 'str'>
```

2．创建空元组

在 Python 中，也可以创建空元组，例如，要创建一个名称为 emptytuple 的空元组，可以使用下面的代码：

```
emptytuple = ()
```

空元组可以应用在为函数传递一个空值或者返回空值时。例如，定义一个函数必须传递一个元组类型的值，而我们还不想为它传递一组数据，那么就可以创建一个空元组传递给它。

3．创建数值元组

在 Python 中，可以使用 tuple()函数直接将 range()函数循环出来的结果转换为数值元组。

tuple()函数的基本语法如下：

```
tuple(data)
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

其中，data 表示可以转换为元组的数据，其类型可以是 range 对象、字符串、元组或者其他可迭代类型的数据。

例如，创建一个 10～20 之间（不包括 20）所有偶数的元组，可以使用下面的代码：

```
tuple(range(10, 20, 2))
```

运行上面的代码，将得到下面的元组：

```
(10, 12, 14, 16, 18)
```

> **说明：**使用 tuple()函数不仅能通过 range 对象创建元组，还可以通过其他对象创建元组。

4．删除元组

对于已经创建的元组，不再使用时，可以使用 del 语句将其删除。语法格式如下：

```
del tuplename
```

其中，tuplename 为要删除元组的名称。

> **说明：**del 语句在实际开发时并不常用。因为 Python 自带的垃圾回收机制会自动销毁不用的元组，所以即使我们不手动将其删除，Python 也会自动将其回收。

例如，定义一个名称为 verse 的元组，然后再用 del 语句将其删除，可以使用下面的代码：

```
01 verse = ("春眠不觉晓","Python 不得了","夜来爬数据","好评知多少")
02 del verse
```

5.3.2　访问元组元素

在 Python 中，如果想将元组的内容输出也比较简单，可以直接使用 print()函数。例如，想打印上面元组中的 untitle 元组，则可以使用下面的代码：

```
print(untitle)
```

执行结果如下：

```
('Python', 28, ('人生苦短', '我用 Python'), ['爬虫', '自动化运维', '云计算', 'Web 开发'])
```

从上面的执行结果中可以看出，在输出元组时，是包括左右两侧的小括号的。如果不想输出全部的元素，也可以通过元组的索引获取指定的元素。例如，要获取元组 untitle 中索引为 0 的元素，可以使用下面的代码：

```
print(untitle[0])
```

执行结果如下：

```
Python
```

从上面的执行结果中可以看出，在输出单个元组元素时，不包括小括号。如果是字符串，还不包括左右的引号。

另外，对于元组也可以采用切片方式获取指定的元素。例如，要访问元组 untitle 中前 3 个元素，可以使用下面的代码：

```
print(untitle[:3])
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

执行结果如下：

```
('Python', 28, ('人生苦短', '我用 Python'))
```

同列表一样，元组也可以使用 for 循环进行遍历。下面通过一个具体的实例演示如何通过 for 循环遍历元组。

5.3.3 修改元组元素

场景描述：在伊米咖啡馆，因为麝香猫咖啡断货，所以店长想要把它换成拿铁咖啡。

实例 01　替换麝香猫咖啡为拿铁咖啡

在 IDLE 中创建一个名称为 cafe_replace.py 的文件，然后在该文件中，定义一个包含 6 个元素的元组，内容为伊米咖啡馆里的咖啡名称，然后修改其中的第 5 个元素的内容为“拿铁”，代码如下：

```
01 coffeename = ('蓝山','卡布奇诺','曼特宁','摩卡','麝香猫','哥伦比亚')  # 定义元组
02 coffeename[4] = '拿铁'              # 将“麝香猫”替换为“拿铁”
03 print(coffeename)
```

运行结果如图 5.12 所示。

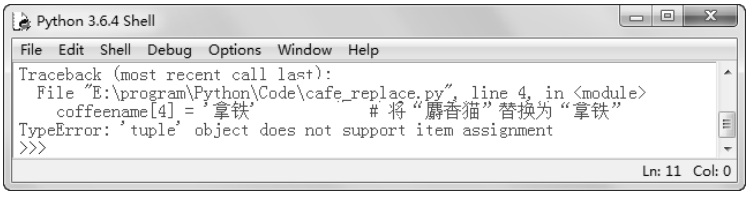


图 5.12　替换麝香猫咖啡为拿铁咖啡出现异常

元组是不可变序列，所以我们不能对它的单个元素值进行修改。但是元组也不是完全不能修改，我们可以对元组进行重新赋值。例如，下面的代码是允许的：

```
01 coffeename = ('蓝山','卡布奇诺','曼特宁','摩卡','麝香猫','哥伦比亚')  # 定义元组
# 对元组进行重新赋值
02 coffeename = ('蓝山','卡布奇诺','曼特宁','摩卡','拿铁','哥伦比亚')
03 print("新元组",coffeename)
```

执行结果如下：

```
新元组 ('蓝山', '卡布奇诺', '曼特宁', '摩卡', '拿铁', '哥伦比亚')
```

从上面的执行结果中可以看出，元组 coffeename 的值已经改变。

另外，还可以对元组进行连接组合。例如，可以使用下面的代码实现在已经存在的元组结尾处添加一个新元组。

```
01 ukguzheng = ('蓝山','卡布奇诺','曼特宁','摩卡')
02 print("原元组：",ukguzheng)
03 ukguzheng = ukguzheng + ('麝香猫','哥伦比亚')
04 print("组合后：",ukguzheng)
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

执行结果如下：

```
原元组： ('蓝山', '卡布奇诺', '曼特宁', '摩卡')
组合后： ('蓝山', '卡布奇诺', '曼特宁', '摩卡', '麝香猫', '哥伦比亚')
```

> **注意：**在进行元组连接时，连接的内容必须都是元组。不能将元组和字符串或者列表进行连接。例如，下面的代码就是错误的：
>
> ```
> 01 ukguzheng = '蓝山','卡布奇诺','曼特宁','摩卡')
> 02 ukguzheng = ukguzheng + ['麝香猫','哥伦比亚']
> ```

常见错误

在进行元组连接时，如果要连接的元组只有一个元素时，一定不要忘记后面的逗号。例如，使用下面的代码将产生如图 5.13 所示的错误。

```
01 ukguzheng = ('蓝山','卡布奇诺','曼特宁','摩卡')
02 ukguzheng = ukguzheng + ('麝香猫')
```

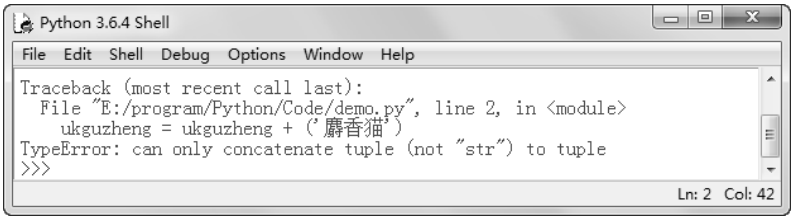


图 5.13　在进行元组连接时产生的异常

5.3.4　元组推导式

使用元组推导式可以快速生成一个元组，它的表现形式和列表推导式类似，只是将列表推导式中的中括号“[]”修改为小括号“()”。例如，我们可以使用下面的代码生成一个包含 10 个随机数的元组。

```
01 import random                                    # 导入 random 标准库
02 randomnumber = (random.randint(10,100) for i in range(10))
03 print("生成的元组为：",randomnumber)
```

执行结果如下：

```
生成的元组为： <generator object <genexpr> at 0x0000000003056620>
```

从上面的执行结果中可以看出，使用元组推导式生成的结果并不是一个元组或者列表，而是一个生成器对象，这一点和列表推导式是不同的。要使用该生成器对象可以将其转换为元组或者列表。其中，转换为元组使用 tuple()函数，而转换为列表则使用 list()函数。

例如，使用元组推导式生成一个包含 10 个随机数的生成器对象，然后将其转换为元组并输出，可以使用下面的代码：

```
01 import random                                    # 导入 random 标准库
02 randomnumber = (random.randint(10,100) for i in range(10))
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
03  randomnumber = tuple(randomnumber)          # 转换为元组
04  print("转换后：",randomnumber)
```

执行结果如下：

```
转换后： (76, 54, 74, 63, 61, 71, 53, 75, 61, 55)
```

要使用通过元组推导器生成的生成器对象，还可以直接通过 for 循环遍历或者直接使用__next()__方法进行遍历。

说明：在 Python 2.x 中，__next__()方法为 next()，也是用于遍历生成器对象的。

例如，通过生成器推导式生成一个包含 3 个元素的生成器对象 number，然后调用 3 次__next__()方法输出每个元素，再将生成器对象 number 转换为元组输出，代码如下：

```
01  number = (i for i in range(3))
02  print(number.__next__())        # 输出第1个元素
03  print(number.__next__())        # 输出第2个元素
04  print(number.__next__())        # 输出第3个元素
05  number = tuple(number)          # 转换为元组
06  print("转换后：",number)
```

运行上面的代码，将显示以下结果：

```
0
1
2
转换后： ()
```

再例如，通过生成器推导式生成一个包括 4 个元素的生成器对象 number，然后应用 for 循环遍历该生成器对象，并输出每一个元素的值，最后再将其转换为元组输出，代码如下：

```
01  number = (i for i in range(4))      # 生成生成器对象
02  for i in number:                    # 遍历生成器对象
03      print(i,end=" ")                # 输出每个元素的值
04  print(tuple(number))                # 转换为元组输出
```

执行结果如下：

```
0 1 2 3 ()
```

从上面的两个示例中可以看出，无论通过哪种方法遍历，如果还想再使用该生成器对象，都必须重新创建一个生成器对象，因为遍历后原生成器对象已经不存在了。

5.3.5 元组与列表的区别

元组和列表都属于序列，它们都可以按照特定顺序存放一组元素，而且类型不受限制，只要是 Python 支持的类型都可以。那么它们之间有什么区别呢？

简单理解：列表类似于我们用铅笔在纸上写下自己喜欢的歌曲，写错了还可以擦掉。而元组则类似于用钢笔写下的歌曲名字，写上了就擦不掉了，除非换一张纸重写。

列表和元组的区别主要体现在以下几个方面。

電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- 列表属于可变序列，它的元素可以随时修改或者删除，而元组属于不可变序列，其中的元素不可以修改，除非整体替换。
- 列表可以使用 append()、extend()、insert()、remove()和 pop()等方法实现添加和修改列表元素，而元组则没有这几个方法，因为不能向元组中添加和修改元素，同样，也不能删除元素。
- 列表可以使用切片访问和修改列表中的元素。元组也支持切片，但是它只支持通过切片访问元组中的元素，不支持修改。
- 元组比列表的访问和处理速度快。所以如果只需要对其中的元素进行访问，而不进行任何修改，建议使用元组而不使用列表。
- 列表不能作为字典的键，而元组则可以。

5.4　字典（dictionary）

字典和列表类似，也是可变序列，不过与列表不同，它是无序的可变序列，保存的内容是以“键—值对”的形式存放的，这类似于我们的新华字典，它可以把拼音和汉字关联起来，通过音节表可以快速找到想要的汉字。其中新华字典里的音节表相当于键（key），而对应的汉字，相当于值（value）。键是唯一的，而值可以有多个。字典在定义一个包含多个命名字段的对象时很有用。

说明：Python 中的字典相当于 Java 或者 C++中的 Map 对象。

字典的主要特征如下。

- 通过键而不是通过索引来读取。

字典有时也称为关联数组或者散列表（hash），它是通过键将一系列的值联系起来的，这样就可以通过键从字典中获取指定项，但不能通过索引来获取。

- 字典是任意对象的无序集合。

字典是无序的，各项是从左到右随机排序的，即保存在字典中的项没有特定的顺序，这样可以提高查找顺序。

- 字典是可变的，并且可以任意嵌套。

字典可以在原处增长或者缩短（无须生成一份副本），并且它支持任意深度的嵌套（它的值可以是列表或者其他的字典）。

- 字典中的键必须唯一。

不允许同一个键出现两次，如果出现两次，则后一个值会被记住。

- 字典中的键必须不可变。

字典中的键是不可变的，所以可以使用数字、字符串或者元组，但不能使用列表。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

5.4.1 字典的创建和删除

定义字典时，每个元素都包含两个部分“键”和“值”。以水果名称和价钱的字典为例，键为水果名称，值为水果价格，如图 5.14 所示。

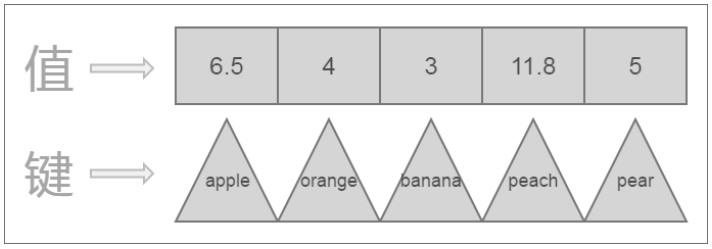


图 5.14 字典示意图

创建字典时，在“键”和“值”之间使用冒号分隔，相邻两个元素使用逗号分隔，所有元素放在一个大括号“{}”中。语法格式如下：

```
dictionary = {'key1':'value1', 'key2':'value2', …, 'keyn':'valuen',}
```

参数说明如下。

- dictionary：表示字典名称。
- key1、key2…keyn：表示元素的键，必须是唯一的，并且不可变，例如可以是字符串、数字或者元组。
- value1、value2…valuen：表示元素的值，可以是任何数据类型，不是必须唯一。

例如，创建一个保存通讯录信息的字典，可以使用下面的代码：

```
01 dictionary = {'qq':'84978981','明日科技':'84978982','无语':'0431-84978981'}
02 print(dictionary)
```

执行结果如下：

```
{'qq': '84978981', 'mr': '84978982', '无语': '0431-84978981'}
```

同列表和元组一样，也可以创建空字典。在 Python 中，可以使用下面两种方法创建空字典：

```
dictionary = {}
```

或者

```
dictionary = dict()
```

Python 的 dict()方法除了可以创建一个空字典外，还可以通过已有数据快速创建字典。主要表现为以下两种形式。

1．通过映射函数创建字典

语法如下：

```
dictionary = dict(zip(list1,list2))
```

参数说明如下。

- dictionary：表示字典名称。
- zip()函数：用于将多个列表或元组对应位置的元素组合为元组，并返回包含这些内容的

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

zip 对象。如果想得到元组，可以将 zip 对象使用 tuple()函数转换元组；如果想得到列表，则可以使用 list()函数将其转换为列表。

说明：在 Python 2.x 中，zip()函数返回的内容为包含元组的列表。

- list1：一个列表，用于指定要生成字典的键。
- list2：一个列表，用于指定要生成字典的值。如果 list1 和 list2 的长度不同，则与最短的列表长度相同。

实例 02　名字与星座对应

场景模拟：某大学的寝室里住着 4 位清秀可人的美女，她们的名字保存在一个列表中，另外，她们每个人的星座对应保存在另一个列表中。

在 IDLE 中创建一个名称为 sign_create.py 的文件，然后在该文件中定义两个包括 4 个元素的列表，再应用 dict()函数和 zip()函数将前两个列表转换为对应的字典，并且输出该字典，代码如下：

```
01 name = ['绮梦','冷伊一','香凝','黛兰']          # 作为键的列表
02 sign = ['水瓶座','射手座','双鱼座','双子座']    # 作为值的列表
03 dictionary = dict(zip(name,sign))             # 转换为字典
04 print(dictionary)                             # 输出转换后的字典
```

运行实例后，将显示如图 5.15 所示的结果。

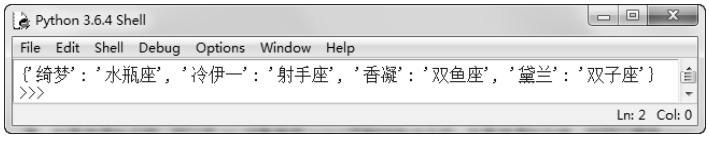


图 5.15　创建字典

2．通过给定的“键—值对”创建字典

语法如下：

```
dictionary = dict(key1=value1,key2=value2,…,keyn=valuen)
```

参数说明如下。

- dictionary：表示字典名称。
- key1、key2…keyn：表示元素的键，必须是唯一的，并且不可变，例如可以是字符串、数字或者元组。
- value1、value2…valuen：表示元素的值，可以是任何数据类型，不是必须唯一。

例如，将上述实例中的名字和星座通过“键—值对”的形式创建一个字典，可以使用下面的代码：

```
01 dictionary =dict(绮梦 = '水瓶座', 冷伊一 = '射手座', 香凝 = '双鱼座', 黛兰 = '双子座')
02 print(dictionary)
```

在 Python 中，还可以使用 dict 对象的 fromkeys()方法创建值为空的字典，语法如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
dictionary = dict.fromkeys(list1)
```

参数说明如下。

- dictionary：表示字典名称。
- list1：表示字典的键的列表。

例如，创建一个只包括名字的字典，可以使用下面的代码：

```
01  name_list = ['绮梦','冷伊一','香凝','黛兰']       # 作为键的列表
02  dictionary = dict.fromkeys(name_list)
03  print(dictionary)
```

执行结果如下：

```
{'绮梦': None, '冷伊一': None, '香凝': None, '黛兰': None}
```

另外，还可以通过已经存在的元组和列表创建字典。例如，下面的代码，创建一个保存名字的元组和保存星座的列表，通过它们创建一个字典，可以使用下面的代码：

```
01  name_tuple = ('绮梦','冷伊一', '香凝', '黛兰')    # 作为键的元组
02  sign = ['水瓶座','射手座','双鱼座','双子座']       # 作为值的列表
03  dict1 = {name_tuple:sign}                        # 创建字典
04  print(dict1)
```

执行结果如下：

```
{('绮梦', '冷伊一', '香凝', '黛兰'): ['水瓶座', '射手座', '双鱼座', '双子座']}
```

如果将作为键的元组修改为列表，再创建一个字典，代码如下：

```
01  name_list = ['绮梦','冷伊一', '香凝', '黛兰'[     # 作为键的元组
02  sign = ['水瓶座','射手座','双鱼座','双子座']       # 作为值的列表
03  dict1 = {name_list:sign}                         # 创建字典
04  print(dict1)
```

执行结果将产生异常，如图 5.16 所示。

```
Traceback (most recent call last):
  File "E:\program\Python\Code\test.py", line 16, in <module>
    dict1 = {name_list:sign}        # 创建字典
TypeError: unhashable type: 'list'
>>>
```

图 5.16　将列表作为字典的键产生的异常

同列表和元组一样，不再需要的字典也可以使用 del 命令删除。例如，通过下面的代码即可将已经定义的字典删除：

```
del dictionary
```

另外，如果只想删除字典的全部元素，可以使用字典对象的 clear()方法实现。执行 clear()方法后，原字典将变为空字典。例如，下面的代码将清除字典的全部元素：

```
dictionary.clear()
```

除了上面介绍的方法可以删除字典元素外，还可以使用字典对象的 pop()方法删除并返回指定“键”的元素，以及使用字典对象的 popitem()方法删除并返回字典中的一个元素。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

5.4.2　通过键-值对访问字典

在 Python 中，如果想将字典的内容输出也比较简单，可以直接使用 print()函数。例如，想打印实例 02 中定义的 dictionary 字典，则可以使用下面的代码：

```
print(dictionary)
```

执行结果如下：

```
{'绮梦': '水瓶座', '冷伊一': '射手座', '香凝': '双鱼座', '黛兰': '双子座'}
```

但是，在使用字典时，很少直接输出它的内容。一般需要根据指定的键得到相应的结果。在 Python 中，访问字典的元素可以通过下标的方式实现，与列表和元组不同，这里的下标不是索引号，而是键。例如，想要获取“冷伊一”的星座，可以使用下面的代码：

```
print(dictionary['冷伊一'])
```

执行结果如下：

```
射手座
```

在使用该方法获取指定键的值时，如果指定的键不存在，将抛出如图 5.17 所示的异常。

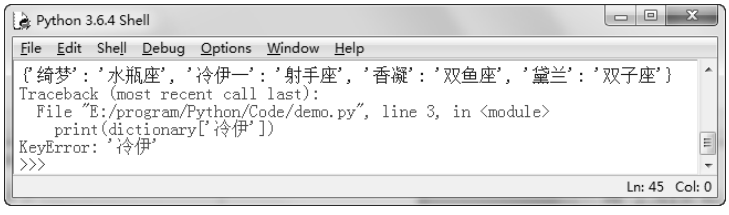


图 5.17　获取指定键不存在时抛出异常

而在实际开发中，很可能我们不知道当前存在什么键，所以需要避免该异常的产生，具体的解决方法是使用 if 语句对不存在的情况进行处理，即给一个默认值。例如，可以将上面的代码修改为以下内容：

```
print("冷伊的星座是:",dictionary['冷伊'] if '' in dictionary else '我的字典里没有此人')
```

当“冷伊”不存在时，将显示以下内容：

```
冷伊的星座是: 我的字典里没有此人
```

另外，Python 中推荐的方法是使用字典对象的 get()方法获取指定键的值，其语法格式如下：

```
dictionary.get(key,[default])
```

其中，dictionary 为字典对象，即要从中获取值的字典；key 为指定的键；default 为可选项，用于指定当指定的“键”不存在时，返回一个默认值，如果省略，则返回 None。

例如，通过 get()方法获取“冷伊一”的星座，可以使用下面的代码：

```
print("冷伊一的星座是:",dictionary.get('冷伊一'))
```

执行结果如下：

```
冷伊一的星座是: 射手座
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

说明：为了解决在获取指定键的值时，因不存在该键而导致抛出异常，可以为 get()方法设置默认值，这样当指定的键不存在时，得到的结果就是指定的默认值。例如，将上面的代码修改为以下内容：

```
print("冷伊的星座是：",dictionary.get('冷伊','我的字典里没有此人'))
```

将得到以下结果：

```
冷伊的星座是： 我的字典里没有此人
```

5.4.3 遍历字典

字典是以键-值对的形式存储数据的，所以就可能需要获取这些键-值对。Python 提供了遍历字典的方法，通过遍历可以获取字典中的全部键-值对。

使用字典对象的 items()方法可以获取字典的键-值对列表，其语法格式如下：

```
dictionary.items()
```

其中，dictionary 为字典对象，返回值为可遍历的键-值对的元组列表。要想获取具体的键-值对，可以通过 for 循环遍历该元组列表。

例如，定义一个字典，然后通过 items()方法获取键-值对的元组列表，并输出全部键-值对，代码如下：

```
01 dictionary = {'qq':'84978981','明日科技':'84978982','无语':'0431-84978981'}
02 for item in dictionary.items():
03     print(item)
```

执行结果如下：

```
('qq', '84978981')
('明日科技', '84978982')
('无语', '0431-84978981')
```

上面的示例得到的是元组中的各个元素，如果想要获取具体的每个键和值，可以使用下面的代码进行遍历：

```
01 dictionary = {'qq':'4006751066','明日科技':'0431-84978982','无语':'0431-84978981'}
02 for key,value in dictionary.items():
03     print(key,"的联系电话是",value)
```

执行结果如下：

```
qq 的联系电话是 4006751066
明日科技 的联系电话是 0431-84978982
无语 的联系电话是 0431-84978981
```

说明：在 Python 中，字典对象还提供了 values()和 keys()方法，用于返回字典的“值”和“键”列表，它们的使用方法同 items()方法类似，也需要通过 for 循环遍历该字典列表，获取对应的值和键。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

5.4.4　添加、修改和删除字典元素

由于字典是可变序列，所以可以随时在其中添加键–值对，这和列表类似。向字典中添加元素的语法格式如下：

```
dictionary[key] = value
```

参数说明如下。

- dictionary：表示字典名称。
- key：表示要添加元素的键，必须是唯一的，并且不可变，例如可以是字符串、数字或者元组。
- value：表示元素的值，可以是任何数据类型，不是必须唯一。

例如，还是以之前的保存 4 位美女星座的场景为例，在创建的字典中添加一个元素，并显示添加后的字典，代码如下：

```
01 dictionary =dict((('绮梦', '水瓶座'),('冷伊一','射手座'), ('香凝','双鱼座'),
('黛兰','双子座')))
02 dictionary["碧琦"] = "巨蟹座"   # 添加一个元素
03 print(dictionary)
```

执行结果如下：

```
{'绮梦': '水瓶座', '冷伊一': '射手座', '香凝': '双鱼座', '黛兰': '双子座', '碧琦': '巨
蟹座'}
```

从上面的结果中可以看出，又添加了一个键为“碧琦”的元素。

由于在字典中，键必须是唯一的，所以如果新添加元素的键与已经存在的键重复，那么将使用新的值替换原来该键的值，这也相当于修改字典的元素。例如，再添加一个键为“香凝”的元素，这次设置她为“天蝎座”，可以使用下面的代码：

```
01 dictionary =dict((('绮梦', '水瓶座'),('冷伊一','射手座'), ('香凝','双鱼座'),
('黛兰','双子座')))
02 dictionary["香凝"] = "天蝎座"   # 添加一个元素。当元素存在时，相当于修改功能
03 print(dictionary)
```

执行结果如下：

```
{'绮梦': '水瓶座', '冷伊一': '射手座', '香凝': '天蝎座', '黛兰': '双子座'}
```

从上面的结果中可以看出，并没有添加一个新的键“香凝”，而是直接对“香凝”进行了修改。

当字典中的某一个元素不需要时，可以使用 del 命令将其删除。例如，要删除字典 dictionary 的键为“香凝”的元素，可以使用下面的代码：

```
01 dictionary =dict((('绮梦', '水瓶座'),('冷伊一','射手座'), ('香凝','双鱼座'),
('黛兰','双子座')))
02 del dictionary["香凝"]    # 删除一个元素
03 print(dictionary)
```

执行结果如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
{'绮梦': '水瓶座', '冷伊一': '射手座', '黛兰': '双子座'}
```

从上面的结果中可以看出，在字典 dictionary 中只剩下 3 个元素了。

注意：当删除一个不存在的键时，将抛出如图 5.18 所示的异常。

```
Traceback (most recent call last):
  File "E:\program\Python\Code\test.py", line 7, in <module>
    del dictionary["香凝1"]    # 删除一个元素
KeyError: '香凝1'
>>>
```

图 5.18　删除一个不存在的键时抛出的异常

因此，需要将上面的代码修改为以下内容，从而防止删除不存在的元素时抛出异常：

```
01 dictionary =dict((('绮梦', '水瓶座'),('冷伊一','射手座'), ('香凝','双鱼座'),
('黛兰','双子座')))
02 if "香凝 1" in dictionary:       # 如果存在
03     del dictionary["香凝 1"]     # 删除一个元素
04 print(dictionary)
```

5.4.5　字典推导式

使用字典推导式可以快速生成一个字典，它的表现形式和列表推导式类似。例如，我们可以使用下面的代码生成一个包含 4 个随机数的字典，其中字典的键使用数字表示：

```
01 import random                                    #导入 random 标准库
02 randomdict = {i:random.randint(10,100) for i in range(1,5)}
03 print("生成的字典为：",randomdict)
```

执行结果如下：

```
生成的字典为： {1: 21, 2: 85, 3: 11, 4: 65}
```

5.5　列表、元组和字典的区别

在本章的 5.2～5.4 节介绍了序列中的列表、元组和字典的应用，下面通过表 5.2 对这几个数据序列进行比较。

表 5.2　列表、元组和字典的区别

数据结构	是否可变	是否有序	定义符号
列表	可变	有序	[]
元组	不可变	有序	()
字典	可变	无序	{key:value}

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

5.6　快学快用

（1）千年虫问题。

“千年虫”问题，也称计算机 2000 年问题，或“千年危机”，是指某些计算机程序在设计时只采用两位十进制数记录年份的最后两位，如 1998 年被表示为“98”、2000 年被表示为“00”，因此当时间跨入 2000 年时，计算机计时系统会将 2000 年解释为 1900 年，造成各种各样的系统功能紊乱，甚至发生灾难性的后果。因此从根本上说千年虫是一种程序处理日期上的 bug，而非病毒。为了解决 2000 年问题，世界各国投入了大量人力、物力，据统计耗资超过 6000 亿美元，这是人类有史以来最昂贵的救灾行动。

下面的序列保存了 10 名 1920—2000 年出生人员的出生年份，为了避免出现千年虫问题，请编写一个小程序，把序列信息中存在千年虫问题的数据进行提取并修改，然后按年龄从高到低输出。

人员序列信息：【45，89，1998，00，75，33，1968，37，1958，90】

输出参考：【1933，1937，1945，1958，1968，1975，1989，1998，1990，2000】

（2）创建一个包含 1～50 之间所有素数的列表，打印显示该列表；随后只保留该列表前 5 个数字，删除其余内容并打印输出相应结果；再将每个元素值加上 20，显示列表内容。

（3）Python 提供了很多有用的内置函数，大家熟悉一下这些函数的用途，在实际开发时如果遇到相关应用，可以联想到相关函数，然后详细学习并使用，事半功倍。逻辑判断内置函数如表 5.3 所示。如果要详细了解这些函数的使用，可以扫描二维码，到快用云端支持平台学习、训练、实战。

表 5.3　逻辑判断内置函数

all(iterable)	• 集合中的元素都为真的时候为真 • 特别的，若为空串返回为 True
any(iterable)	• 集合中的元素有一个为真的时候为真 • 特别的，若为空串返回为 False
cmp(x, y)	x < y，返回负数；x == y，返回 0；x > y，返回正数

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 6 章

字符串及正则表达式

字符串几乎是所有编程语言在项目开发过程中涉及最多的内容。大部分项目的运行结果都需要以文本的形式展示给客户，比如财务系统的总账报表，电子游戏的比赛结果，火车站的列车时刻表等，这些都需要经过程序精密的计算、判断和梳理，然后将我们想要的内容用文本的形式直观地展示出来。曾经有一位“久经沙场”的老程序员说过一句话：“开发一个项目，基本上就是在不断地处理字符串。”

6.1 字符串常用操作

在 Python 开发过程中，为了实现某项功能，经常需要对某些字符串进行特殊处理，如拼接字符串、截取字符串、格式化字符串等。下面将对 Python 中常用的字符串操作方法进行介绍。

6.1.1 拼接字符串

使用“+”运算符可完成对多个字符串的拼接，“+”运算符可以连接多个字符串并产生一个字符串对象。

例如，定义两个字符串，一个保存英文版的名言，另一个保存中文版的名言，然后使用“+”运算符连接，代码如下：

```
01 mot_en = 'Remembrance is a form of meeting. Frgetfulness is a form of freedom.'
02 mot_cn = '记忆是一种相遇。遗忘是一种自由。'
03 print(mot_en + '——' + mot_cn)
```

字符串不允许直接与其他类型的数据进行拼接，例如，使用下面的代码将字符串与数值拼接在一起，将产生如图 6.1 所示的异常。

```
01 str1 = '我今天一共走了'          # 定义字符串
02 num = 12098                    # 定义一个整数
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
03  str2 = '步'                          # 定义字符串
04  print(str1 + num + str2)             # 对字符串和整数进行拼接
```

```
Traceback (most recent call last):
  File "E:\program\Python\Code\test.py", line 19, in <module>
    print(str1 + num + str2)
TypeError: must be str, not int
>>>
```

图 6.1　字符串和整数拼接抛出的异常

解决该问题，可以使用 str()函数将整数转换为字符串，修改后的代码如下：

```
01  str1 = '今天我一共走了'                 # 定义字符串
02  num = 12098                            # 定义一个整数
03  str2 = '步'                            # 定义字符串
04  print(str1 + str(num) + str2)          # 对字符串和整数进行拼接
```

执行上面的代码，将显示以下内容：

```
今天我一共走了12098步
```

6.1.2　计算字符串的长度

由于不同的字符所占字节数不同，所以要计算字符串的长度需要先了解各字符所占的字节数。在 Python 中，数字、英文、小数点、下画线和空格占 1 字节；一个汉字可能会占 2～4 字节，占几字节取决于采用的编码。汉字在 GBK/GB2312 编码中占 2 字节，在 UTF-8/Unicode 中一般占 3 字节（或 4 字节）。下面以 Python 默认的 UTF-8 编码为例进行说明，即一个汉字占 3 字节，如图 6.2 所示。

图 6.2　汉字和英文所占字节数

在 Python 中，提供了 len()函数计算字符串的长度，语法格式如下：

```
len(string)
```

其中，string 用于指定要进行长度统计的字符串。

例如，定义一个字符串，内容为“人生苦短，我用 Python!”，然后应用 len()函数计算该字符串的长度，代码如下：

```
01  str1 = '人生苦短，我用 Python!'     # 定义字符串
02  length = len(str1)                 # 计算字符串的长度
03  print(length)
```

上面的代码在执行后，将显示“14”。

从上面的结果中可以看出，在默认情况下，通过 len()函数计算字符串的长度时，不区分英文、数字和汉字，所有字符都认为是一个字节。

在实际开发时，有时需要获取字符串实际所占的字节数，即如果采用 UTF-8 编码，汉字

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

占 3 字节，采用 GBK 或者 GB2312 编码时，汉字占 2 字节。这时，可以通过使用 encode()方法编码后再进行获取。例如，如果要获取采用 UTF-8 编码的字符串长度，可以使用下面的代码：

```
01 str1 = '人生苦短，我用 Python!'        # 定义字符串
02 length = len(str1.encode())          # 计算 UTF-8 编码的字符串长度
03 print(length)
```

上面的代码在执行后，将显示“28”。这是因为汉字加中文标点符号共 7 个，占 21 字节，英文字母和英文标点符号占 7 字节，共 28 字节。

如果要获取采用 GBK 编码的字符串长度，可以使用下面的代码：

```
01 str1 = '人生苦短，我用 Python!'        # 定义字符串
02 length = len(str1.encode('gbk'))     # 计算 GBK 编码的字符串长度
03 print(length)
```

上面的代码在执行后，将显示“21”。这是因为汉字加中文标点符号共 7 个，占 14 字节，英文字母和英文标点符号占 7 字节，共 21 字节。

6.1.3 截取字符串

由于字符串也属于序列，所以要截取字符串，可以采用切片方法实现。通过切片方法截取字符串的语法格式如下：

```
string[start : end : step]
```

参数说明如下。

- string：表示要截取的字符串。
- start：表示要截取的第一个字符的索引（包括该字符），如果不指定，则默认为 0。
- end：表示要截取的最后一个字符的索引（不包括该字符），如果不指定则默认为字符串的长度。
- step：表示切片的步长，如果省略，则默认为 1，当省略该步长时，最后一个冒号也可以省略。

> **说明：**字符串的索引同序列的索引是一样的，也是从 0 开始的，并且每个字符占一个位置，如图 6.3 所示。

图 6.3　字符串的索引示意图

例如，定义一个字符串，然后应用切片方法截取不同长度的子字符串并输出，代码如下：

```
01 str1 = '人生苦短，我用 Python!'    # 定义字符串
02 substr1 = str1[1]                # 截取第 2 个字符
03 substr2 = str1[5:]               # 从第 6 个字符截取
04 substr3 = str1[:5]               # 从左边开始截取 5 个字符
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
substr4 = str1[2:5]              # 截取第 3 个到第 5 个字符
print('原字符串: ',str1)
print(substr1 + '\n' + substr2 + '\n' + substr3 + '\n' + substr4)
```

上面的代码执行后，将显示以下内容：

```
原字符串:  人生苦短，我用 Python!
生
我用 Python!
人生苦短，
苦短，
```

> **注意：**在进行字符串截取时，如果指定的索引不存在，则会抛出如图 6.4 所示的异常，要解决该问题，可以采用 try…except 语句捕获异常。例如，下面的代码在执行后将不抛出异常：

```
str1 = '人生苦短，我用 Python!'    # 定义字符串
try:
    substr1 = str1[15]           # 截取第 15 个字符
except IndexError:
    print('指定的索引不存在')
```

```
Traceback (most recent call last):
  File "E:\program\Python\Code\test.py", line 19, in <module>
    substr1 = str1[15]    # 截取第15个字符
IndexError: string index out of range
>>>
```

图 6.4　指定的索引不存在时抛出的异常

6.1.4　分隔字符串

在 Python 中，字符串对象的 split()方法可以实现字符串的分隔，即把一个字符串按照指定的分隔符切分为字符串列表，该列表的元素中不包括分隔符。split()方法的语法格式如下：

```
str.split(sep, maxsplit)
```

参数说明如下。

- str：表示要进行分隔的字符串。
- sep：用于指定分隔符，可以包含多个字符，默认为 None，即所有空字符（包括空格、换行符“\n”、制表符“\t”等）。
- maxsplit：可选参数，用于指定分隔的次数，如果不指定或者为-1，则分隔次数没有限制，否则返回结果列表的元素个数最多为 maxsplit+1。

> **说明：**在 split()方法中，如果不指定 sep 参数，那么也不能指定 maxsplit 参数。

- 返回值：分隔后的字符串列表。

例如，定义一个保存明日学院网址的字符串，然后使用 split()方法根据不同的分隔符进行分隔，代码如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
str1 = '明 日 学 院 官 网  >>>  www.mingrisoft.com'
print('原字符串：',str1)
list1 = str1.split()            # 采用默认分隔符进行分隔
list2 = str1.split('>>>')       # 采用多个字符进行分隔
list3 = str1.split('.')         # 采用.号进行分隔
list4 = str1.split(' ',4)       # 采用空格进行分隔，并且只分隔前 4 个
print(str(list1) + '\n' + str(list2) + '\n' + str(list3) + '\n' + str(list4))
list5 = str1.split('>')         # 采用>号进行分隔
print(list5)
```

上面的代码在执行后，将显示以下内容：

```
原字符串： 明 日 学 院 官 网  >>>  www.mingrisoft.com
['明', '日', '学', '院', '官', '网', '>>>', 'www.mingrisoft.com']
['明 日 学 院 官 网 ', ' www.mingrisoft.com']
['明 日 学 院 官 网  >>>  www', 'mingrisoft', 'com']
['明', '日', '学', '院', '官 网  >>>  www.mingrisoft.com']
['明 日 学 院 官 网  ', '', '', '  www.mingrisoft.com']
```

> **说明：**在使用 split()方法时，如果不指定参数，默认采用空格符进行分隔，这时无论有几个空格，空格符都将作为一个分隔符进行分隔。例如，在上面的示例中，在“网”和“>”之间有两个空格，但是分隔结果（第二行内容）中已经被全部过滤掉了。如果指定一个分隔符，那么当这个分隔符出现多个时，就会每个分隔一次，没有得到内容的，将产生一个空元素。例如，上面结果中的最后一行，就出现了两个空元素。

6.1.5 检索字符串

在 Python 中，字符串对象提供了很多应用于字符串查找的方法，这里主要介绍以下几种方法。

1．count()方法

count()方法用于检索指定字符串在另一个字符串中出现的次数。如果检索的字符串不存在，则返回 0，否则返回出现的次数，其语法格式如下：

```
str.count(sub[, start[, end]])
```

参数说明如下。

- str：表示原字符串。
- sub：表示要检索的子字符串。
- start：可选参数，表示检索范围的起始位置的索引，如果不指定，则从头开始检索。
- end：可选参数，表示检索范围的结束位置的索引，如果不指定，则一直检索到结尾。

例如，定义一个字符串，然后应用 count()方法检索该字符串中“@”符号出现的次数，代码如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
01 str1 = '@明日科技 @扎克伯格 @俞敏洪'
02 print('字符串“',str1,'”中包括',str1.count('@'),'个@符号')
```

上面的代码执行后，将显示以下结果：

```
字符串“ @明日科技 @扎克伯格 @俞敏洪 ”中包括 3 个@符号
```

2．find()方法

该方法用于检索是否包含指定的子字符串。如果检索的字符串不存在，则返回-1，否则返回首次出现该子字符串时的索引，其语法格式如下：

```
str.find(sub[, start[, end]])
```

参数说明如下。

- str：表示原字符串。
- sub：表示要检索的子字符串。
- start：可选参数，表示检索范围的起始位置的索引，如果不指定，则从头开始检索。
- end：可选参数，表示检索范围的结束位置的索引，如果不指定，则一直检索到结尾。

例如，定义一个字符串，然后应用 find()方法检索该字符串中首次出现“@”符号的位置索引，代码如下：

```
01 str1 = '@明日科技 @扎克伯格 @俞敏洪'
02 print('字符串“',str1,'”中@符号首次出现的位置索引为：',str1.find('@'))
```

上面的代码执行后，将显示以下结果：

```
字符串“ @明日科技 @扎克伯格 @俞敏洪 ”中@符号首次出现的位置索引为： 0
```

> **说明：**如果只是想要判断指定的字符串是否存在，可以使用 in 关键字实现。例如，上面的字符串 str1 中是否存在@符号，可以使用 print('@' in str1)，如果存在就返回 True，否则返回 False。另外，也可以根据 find()方法的返回值是否大于-1 来确定是否存在。

如果输入的子字符串在原字符串中不存在，将返回-1，例如下面的代码：

```
01 str1 = '@明日科技 @扎克伯格 @俞敏洪'
02 print('字符串“',str1,'”中@符号首次出现的位置索引为：',str1.find('*'))
```

上面的代码执行后，将显示以下结果：

```
字符串“ @明日科技 @扎克伯格 @俞敏洪 ”中@符号首次出现的位置索引为： -1
```

> **说明：**Python 的字符串对象还提供了 rfind()方法，其作用与 find()方法类似，只是 rfind()方法从右边开始查找。

3．index()方法

index()方法同 find()方法类似，也是用于检索是否包含指定的子字符串。只不过如果使用 index()方法，当指定的字符串不存在时会抛出异常，其语法格式如下：

```
str.index(sub[, start[, end]])
```

参数说明如下。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- str：表示原字符串。
- sub：表示要检索的子字符串。
- start：可选参数，表示检索范围的起始位置的索引，如果不指定，则从头开始检索。
- end：可选参数，表示检索范围的结束位置的索引，如果不指定，则一直检索到结尾。

例如，定义一个字符串，然后应用 index()方法检索该字符串中首次出现“@”符号的位置索引，代码如下：

```
01 str1 = '@明日科技 @扎克伯格 @俞敏洪'
02 print('字符串“',str1,'”中@符号首次出现的位置索引为：',str1.index('@'))
```

上面的代码执行后，将显示以下结果：

```
字符串“ @明日科技 @扎克伯格 @俞敏洪 ”中@符号首次出现的位置索引为： 0
```

如果输入的子字符串在原字符串中不存在，将会产生异常。例如下面的代码：

```
01 str1 = '@明日科技 @扎克伯格 @俞敏洪'
02 print('字符串“',str1,'”中@符号首次出现的位置索引为：',str1.index('*'))
```

上面的代码执行后，将显示如图 6.5 所示的异常。

```
Traceback (most recent call last):
  File "E:\program\Python\Code\test.py", line 7, in <module>
    print('字符串“',str1,'”中@符号首次出现位置索引为：',str1.index('*'))
ValueError: substring not found
>>>
```

图 6.5　index 检索不存在元素时出现的异常

> **说明：**Python 的字符串对象还提供了 rindex()方法，其作用与 index()方法类似，只是 rindex()方法从右边开始查找。

4．startswith()方法

该方法用于检索字符串是否以指定子字符串开头，如果是则返回 True，否则返回 False。语法格式如下：

```
str.startswith(prefix[, start[, end]])
```

参数说明如下。

- str：表示原字符串。
- prefix：表示要检索的子字符串。
- start：可选参数，表示检索范围的起始位置的索引，如果不指定，则从头开始检索。
- end：可选参数，表示检索范围的结束位置的索引，如果不指定，则一直检索到结尾。

例如，定义一个字符串，然后应用 startswith()方法检索该字符串是否以“@”符号开头，代码如下：

```
01 str1 = '@明日科技 @扎克伯格 @俞敏洪'
02 print('判断字符串“',str1,'”是否以@符号开头，结果为：',str1.startswith('@'))
```

上面的代码执行后，将显示以下结果：

```
判断字符串“ @明日科技 @扎克伯格 @俞敏洪 ”是否以@符号开头，结果为： True
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

5．endswith()方法

该方法用于检索字符串是否以指定子字符串结尾，如果是则返回 True，否则返回 False。语法格式如下：

```
str.endswith(suffix[, start[, end]])
```

参数说明如下。

- str：表示原字符串。
- suffix：表示要检索的子字符串。
- start：可选参数，表示检索范围的起始位置的索引，如果不指定，则从头开始检索。
- end：可选参数，表示检索范围的结束位置的索引，如果不指定，则一直检索到结尾。

例如，定义一个字符串，然后应用 endswith()方法检索该字符串是否以“.com”结尾，代码如下：

```
01 str1 = ' http://www.mingrisoft.com'
02 print('判断字符串“',str1,'”是否以.com结尾，结果为：',str1.endswith('.com'))
```

上面的代码执行后，将显示以下结果：

```
判断字符串“ http://www.mingrisoft.com ”是否以.com结尾，结果为： True
```

6.1.6　字母的大小写转换

在 Python 中，字符串对象提供了 lower()方法和 upper()方法进行字母的大小写转换，即用于将大写字母 ABC 转换为小写字母 abc，或者将小写字母 abc 转换为大写字母 ABC，如图 6.6 所示。下面分别进行介绍。

图 6.6　字母大小写转换示意图

1．lower()方法

lower()方法用于将字符串中的全部大写字母转换为小写字母。如果字符串中没有应该被转换的字符，则将原字符串返回；否则将返回一个新的字符串，将原字符串中每个该进行小写转换的字符都转换成等价的小写字符。字符长度与原字符长度相同。lower()方法的语法格式如下：

```
str.lower()
```

其中，str 为要进行转换的字符串。

例如，下面的代码将全部显示为小写字母。

```
01 str1 = 'WWW.Mingrisoft.com'
02 print('原字符串：',str1)
03 print('新字符串：',str1.lower())  # 转换为全部小写输出
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2．upper()方法

upper()方法用于将字符串的全部小写字母转换为大写字母。如果字符串中没有应该被转换的字符，则将原字符串返回；否则返回一个新字符串，将原字符串中每个该进行大写转换的字符都转换成等价的大写字符。新字符长度与原字符长度相同。upper()方法的语法格式如下：

```
str.upper()
```

其中，str 为要进行转换的字符串。

例如，下面的代码将全部显示为大写字母。

```
01 str1 = 'WWW.Mingrisoft.com'
02 print('原字符串：',str1)
03 print('新字符串：',str1.upper())  # 转换为全部大写输出
```

6.1.7 去除字符串中的空格和特殊字符

用户在输入数据时，可能会无意中输入多余的空格，或在一些情况下，字符串前后不允许出现空格和特殊字符，此时就需要去除字符串中的空格和特殊字符。例如，图 6.7 中“HELLO”这个字符串前后都有一个空格。可以使用 Python 中提供的 strip()函数去除字符串左右两边的空格和特殊字符，也可以使用 lstrip()函数去除字符串左边的空格和特殊字符，或使用 rstrip()函数去除字符串中右边的空格和特殊字符。

图 6.7　前后包含空格的字符串

说明：这里的特殊字符是指制表符（\t）、回车符（\r）、换行符（\n）等。

1．strip()方法

strip()方法用于去掉字符串左、右两侧的空格和特殊字符，其语法格式如下：

```
str.strip([chars])
```

其中，str 为要去除空格的字符串；chars 为可选参数，用于指定要去除的字符，可以指定多个，例如设置 chars 为“@.”，则去除左、右两侧包括的@或.。如果不指定 chars 参数，默认将去除空格、制表符（\t）、回车符（\r）、换行符（\n）等。

例如，先定义一个字符串，首尾包括空格、制表符、换行符和回车符等，然后去除空格和这些特殊字符；再定义一个字符串，首尾包括@或.字符，最后去掉@和.，代码如下：

```
01 str1 = ' http://www.mingrisoft.com  \t\n\r'
02 print('原字符串 str1：' + str1 + '。')
03 print('字符串：' + str1.strip() + '。')      # 去除字符串首尾的空格和特殊字符
04 str2 = '@明日科技.@.'
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
05  print('原字符串 str2: ' + str2 + '。')
06  print('字符串: ' + str2.strip('@.') + '。')      # 去除字符串首尾的@或者.
```

上面的代码运行后，将显示如图 6.8 所示的结果。

```
原字符串str1：  http://www.mingrisoft.com
。
字符串：http://www.mingrisoft.com。
原字符串str2：@明日科技.@.。
字符串：明日科技。
>>>
```

图 6.8　strip()方法示例

2．lstrip()方法

lstrip()方法用于去掉字符串左侧的空格和特殊字符，其语法格式如下：

```
str.lstrip([chars])
```

其中，str 为要去除空格的字符串；chars 为可选参数，用于指定要去除的字符，可以指定多个，例如设置 chars 为“@.”，则去除左侧包括的@或.。如果不指定 chars 参数，默认将去除空格、制表符（\t）、回车符（\r）、换行符（\n）等。

例如，先定义一个字符串，左侧包括一个制表符和一个空格，然后去除空格和制表符；再定义一个字符串，左侧包括一个@符号，最后去掉@符号，代码如下：

```
01  str1 = '\t http://www.mingrisoft.com'
02  print('原字符串 str1: ' + str1 + '。')
03  print('字符串: ' + str1.lstrip() + '。')       # 去除字符串左侧的空格和制表符
04  str2 = '@明日科技'
05  print('原字符串 str2: ' + str2 + '。')
06  print('字符串: ' + str2.lstrip('@') + '。')    # 去除字符串左侧的@
```

上面的代码运行后，将显示如图 6.9 所示的结果。

```
原字符串str1：    http://www.mingrisoft.com。
字符串：http://www.mingrisoft.com。
原字符串str2：@明日科技。
字符串：明日科技。
>>>
```

图 6.9　lstrip()方法示例

3．rstrip()方法

rstrip()方法用于去掉字符串右侧的空格和特殊字符，其语法格式如下：

```
str.lstrip([chars])
```

其中，str 为要去除空格的字符串；chars 为可选参数，用于指定要去除的字符，可以指定多个，例如设置 chars 为“@.”，则去除右侧包括的@或 .。如果不指定 chars 参数，默认将去除空格、制表符（\t）、回车符（\r）、换行符（\n）等。

例如，先定义一个字符串，右侧包括一个制表符和一个空格，然后去除空格和制表符；再定义一个字符串，右侧包括一个逗号“,”，最后去掉逗号，代码如下：

```
01  str1 = ' http://www.mingrisoft.com\t '
02  print('原字符串 str1: ' + str1 + '。')
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
print('字符串: ' + str1.rstrip() + '。')       # 去除字符串右侧的空格和制表符
str2 = '明日科技,'
print('原字符串str2: ' + str2 + '。')
print('字符串: ' + str2.rstrip(',') + '。')    # 去除字符串右侧的逗号
```

上面的代码运行后，将显示如图 6.10 所示的结果。

```
原字符串str1：  http://www.mingrisoft.com          。
字符串：  http://www.mingrisoft.com。
原字符串str2：明日科技,。
字符串：明日科技。
>>>
```

图 6.10　rstrip()方法示例

6.1.8　格式化字符串

格式化字符串的意思是先制定一个模板，在这个模板中预留几个空位，然后根据需要填上相应的内容。这些空位需要通过指定的符号标记（也称为占位符），而这些符号还不会显示出来。在 Python 中，格式化字符串有以下两种方法。

1．使用%操作符

在 Python 中，要实现格式化字符串，可以使用“%”操作符，语法格式如下：

```
'%[-][+][0][m][.n]格式化字符'%exp
```

参数说明如下。

- -：可选参数，用于指定左对齐，正数前方无符号，负数前方加负号。
- +：可选参数，用于指定右对齐，正数前方加正号，负数前方加负号。
- 0：可选参数，表示右对齐，正数前方无符号，负数前方加负号，用 0 填充空白处（一般与 m 参数一起使用）。
- m：可选参数，表示占用宽度。
- .n：可选参数，表示小数点后保留的位数。
- 格式化字符：用于指定类型，其值如表 6.1 所示。

表 6.1　常用的格式化字符

格式化字符	说　明	格式化字符	说　明
%s	字符串（采用 str()显示）	%r	字符串（采用 repr()显示）
%c	单个字符	%o	八进制整数
%d 或者%i	十进制整数	%e	指数（基底写为 e）
%x	十六进制整数	%E	指数（基底写为 E）
%f 或者%F	浮点数	%%	字符%

- exp：要转换的项。如果要指定的项有多个，需要通过元组的形式进行指定，但不能使用列表。

例如，格式化输出一个保存公司信息的字符串，代码如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
01 template = '编号: %09d\t 公司名称: %s \t 官网: http://www.%s.com'  # 定义模板
02 context1 = (7,'百度','baidu')                                    # 定义要转换的内容 1
03 context2 = (8,'明日学院','mingrisoft')                           # 定义要转换的内容 2
04 print(template%context1)                                         # 格式化输出
05 print(template%context2)                                         # 格式化输出
```

上面的代码运行后将显示如图 6.11 所示的效果，即按照指定模板格式输出两条公司信息。

```
编号: 000000007 公司名称: 百度        官网: http://www.baidu.com
编号: 000000008 公司名称: 明日学院    官网: http://www.mingrisoft.com
>>>
```

图 6.11　格式化输出公司信息

> **说明：**由于使用%操作符是早期 Python 中提供的方法，自从 Python 2.6 版本开始，字符串对象提供了 format()方法对字符串进行格式化。现在一些 Python 社区也推荐使用这种方法，所以建议大家重点学习 format()方法的使用。

2．使用字符串对象的 format()方法

字符串对象提供了 format()方法用于字符串格式化，其语法格式如下：

```
str.format(args)
```

其中，format 用于指定字符串的显示样式（模板）；args 用于指定要转换的项，如果有多项，则用逗号进行分隔。

下面重点介绍如何创建模板。在创建模板时，需要使用{}和:指定占位符，基本语法格式如下：

```
{[index][:[[fill]align][sign][#][width][.precision][type]]}
```

参数说明如下。

- index：可选参数，用于指定要设置格式的对象在参数列表中的索引位置，索引值从 0 开始。如果省略，则根据值的先后顺序自动分配。

> **说明：**当一个模板中出现多个占位符时，指定索引位置的规范需统一，即全部采用手动指定，或者全部采用自动指定。例如，定义“'我是数值：{:d}，我是字符串：{1:s}'”模板是错误的，会抛出如图 6.12 所示的异常。

```
Traceback (most recent call last):
  File "E:\program\Python\Code\test.py", line 17, in <module>
    print(template.format(7,'明日学院'))
ValueError: cannot switch from automatic field numbering to manual field specification
>>>
```

图 6.12　字段规范不统一抛出的异常

- fill：可选参数，用于指定空白处填充的字符。
- align：可选参数，用于指定对齐方式（值为<表示内容左对齐；值为>表示内容右对齐；值为=表示内容右对齐，将符号放在填充内容的最左侧，且只对数字类型有效；值为^表示内容居中），需要配合 width 一起使用。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- sign：可选参数，用于指定有无符号数（值为+表示正数加正号，负数加负号；值为-表示正数不变，负数加负号；值为空格表示正数加空格，负数加负号）。
- #：可选参数，对于二进制数、八进制数和十六进制数，如果加上#，表示会显示 0b/0o/0x 前缀，否则不显示前缀。
- width：可选参数，用于指定所占宽度。
- .precision：可选参数，用于指定保留的小数位数。
- type：可选参数，用于指定类型，format()方法中常用的格式化字符如表 6.2 所示。

表 6.2　format()方法中常用的格式化字符

格式化字符	说　明	格式化字符	说　明
s	对字符串类型格式化	b	将十进制数自动转换成二进制数表示再格式化
d	十进制数	o	将十进制数自动转换成八进制数表示再格式化
c	将十进制数自动转换成对应的 Unicode 字符	x 或者 X	将十进制数自动转换成十六进制数表示再格式化
e 或者 E	转换为科学计数法表示再格式化	f 或者 F	转换为浮点数（默认小数点后保留 6 位）再格式化
g 或者 G	自动在 e 和 f 或者 E 和 F 中切换	%	显示百分比（默认显示小数点后 6 位）

例如，定义一个保存公司信息的字符串模板，然后应用该模板输出不同公司的信息，代码如下：

```
01 template = '编号：{:0>9s}\t 公司名称： {:s} \t 官网： http://www.{:s}.com'  # 定义模板
02 context1 = template.format('7','百度','baidu')          # 转换内容 1
03 context2 = template.format('8','明日学院','mingrisoft') # 转换内容 2
04 print(context1)                                          # 输出格式化后的字符串
05 print(context2)                                          # 输出格式化后的字符串
```

上面的代码运行后将显示如图 6.13 所示的效果，即按照指定模板格式输出两条公司信息。

```
编号：000000007 公司名称： 百度        官网： http://www.baidu.com
编号：000000008 公司名称： 明日学院    官网： http://www.mingrisoft.com
>>>
```

图 6.13　格式化输出公司信息

在实际开发中，数值类型有多种显示方式，比如货币形式、百分比形式等，使用 format()方法可以将数值格式化为不同的形式。

6.2　正则表达式基础

在处理字符串时，经常会有查找符合某些复杂规则的字符串的需求，正则表达式就是用于描述这些规则的工具。换句话说，正则表达式就是记录文本规则的代码。对于接触过 DOS 的用户来说，如果想匹配当前文件夹下所有的文本文件，可以输入“dir *.txt”命令，按<Enter>键后，所有“.txt”文件会被列出来。这里的“*.txt”可理解为一个简单的正则表达式。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

6.2.1　行定位符

行定位符就是用来描述字符串的边界。“^”表示行的开始；“$”表示行的结尾。如：

```
^tm
```

该表达式表示要匹配字符串 tm 的开始位置是行头，如 tm equal Tomorrow Moon 就可以匹配，而 Tomorrow Moon equal tm 则不匹配。但如果使用：

```
tm$
```

后者可以匹配而前者不能匹配。如果要匹配的字符串可以出现在字符串的任意部分，那么可以直接写成：

```
tm
```

这样两个字符串就都可以匹配了。

6.2.2　元字符

现在我们已经知道几个很有用的元字符了，如^和$。其实，正则表达式里还有很多元字符，下面来看看更多的例子：

```
\bmr\w*\b
```

匹配以字母 mr 开头的单词，先是从某个单词开始处（\b），然后匹配字母 mr，接着是任意数量的字母或数字（\w*），最后单词结束处（\b）。该表达式可以匹配“mrsoft”、“mrbook”和“mr123456”等。更多常用元字符如表 6.3 所示。

表 6.3　常用元字符

代　码	说　明
.	匹配除换行符以外的任意字符
\w	匹配字母、数字、下画线或汉字
\s	匹配任意的空格符
\d	匹配数字
\b	匹配单词的开始或结束
^	匹配字符串的开始
$	匹配字符串的结束

6.2.3　限定符

在上面的例子中，使用\w*匹配任意数量的字母或数字。如果想匹配特定数量的数字，该如何表示呢？正则表达式为我们提供了限定符（指定数量的字符）来实现该功能，如匹配 8 位 QQ 号可用如下方式表示：

```
^\d{8}$
```

常用的限定符如表 6.4 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

表 6.4　常用限定符

限 定 符	说　明	举　例
?	匹配前面的字符零次或一次	colou?r，该表达式可以匹配 colour 和 color
+	匹配前面的字符一次或多次	go+gle，该表达式可以匹配的范围从 gogle 到 goo…gle
*	匹配前面的字符零次或多次	go*gle，该表达式可以匹配的范围从 ggle 到 goo…gle
{n}	匹配前面的字符 *n* 次	go{2}gle，该表达式只匹配 google
{n,}	匹配前面的字符最少 *n* 次	go{2,}gle，该表达式可以匹配的范围从 google 到 goo…gle
{n,m}	匹配前面的字符最少 *n* 次，最多 *m* 次	employe{0,2}，该表达式可以匹配 employ、employe 和 employee 3 种情况

6.2.4　字符类

正则表达式查找数字和字母是很简单的，因为已经有了对应这些字符集合的元字符（如\d，\w），但是如果要匹配没有预定义元字符的字符集合（比如元音字母 a、e、i、o、u），应该怎么办呢？

很简单，只需要在方括号里列出它们就行了，像[aeiou]就匹配任何一个英文元音字母，[.?!]匹配标点符号.、?或!。也可以轻松指定一个字符范围，像[0-9]代表的含意与\d 是完全一致的，即一位数字。同理，[a-z0-9A-Z_]也完全等同于\w（如果只考虑英文的话）。

> **说明：**要想匹配给定字符串中任意一个汉字，可以使用[\u4e00-\u9fa5]；如果要匹配连续多个汉字，可以使用[\u4e00-\u9fa5]+。

6.2.5　排除字符

在 6.2.4 节列出的是匹配符合指定字符集合的字符串，现在反过来，匹配不符合指定字符集合的字符串。正则表达式提供了“^”字符，这个元字符在 6.2.1 节中出现过，表示行的开始。而这里将会放到方括号中，表示排除的意思。例如：

```
[^a-zA-Z]
```

该表达式用于匹配一个不是字母的字符。

6.2.6　选择字符

试想一下，如何匹配身份证号码？首先需要了解一下身份证号码的规则，身份证号码长度为 15 位或者 18 位。如果为 15 位时，则全为数字；如果为 18 位时，前 17 位为数字，最后一位是校验位，可能为数字或字符 X（x）。

在上面的描述中，包含着条件选择的逻辑，这就需要使用选择字符“|”来实现，该字符可以理解为“或”，匹配身份证号码的表达式可以写成如下方式：

```
^\d{15}$)|(^\d{18}$)|(^\d{17}(\d|X|x)$
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

该表达式的意思是匹配 15 位数字、18 位数字，或者 17 位数字和最后一位。最后一位可以是数字或者 X（x）。

6.2.7　转义字符

正则表达式中的转义字符“\”和 Python 中的转义字符大同小异，都是将特殊字符（如“.”、“?”、“\”等）变为普通的字符。举一个 IP 地址的实例，用正则表达式匹配诸如 127.0.0.1 这样格式的 IP 地址。如果直接使用点字符，格式为：

```
[1-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,3}
```

这显然不对，因为“.”可以匹配一个任意字符。这时，不仅是 127.0.0.1 这样的 IP，连 127101011 这样的字符串也会被匹配出来。所以在使用“.”时，需要使用转义字符“\”。修改上面的正则表达式为：

```
[1-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}
```

> **说明：**括号在正则表达式中也算是一个元字符。

6.2.8　分组

通过 6.2.6 节中的例子，相信读者已经对小括号的作用有了一定的了解。小括号的第一个作用就是可以改变限定符的作用范围，如“|”“*”“^”等。下面来看一个表达式：

```
(thir|four)th
```

这个表达式的意思是匹配单词 thirth 或 fourth，如果不使用小括号，那么就变成了匹配单词 thir 和 fourth 了。

小括号的第二个作用是分组，也就是子表达式。如(\.[0–9]{1,3}){3}就是对分组(\.[0–9]{1,3})进行重复操作。

6.2.9　在Python中使用正则表达式语法

在 Python 中使用正则表达式时，是将其作为模式字符串使用的。例如，将匹配不是字母的一个字符的正则表达式表示为模式字符串，可以使用下面的代码：

```
'[^a-zA-Z]'
```

而如果将匹配以字母 m 开头的单词的正则表达式转换为模式字符串，则不能直接在其两侧添加引号定界符，例如，下面的代码是不正确的：

```
'\bm\w*\b'
```

需要将其中的“\”进行转义，转换后的结果为：

```
'\\bm\\w*\\b'
```

由于模式字符串中可能包括大量的特殊字符和反斜杠，所以需要写为原生字符串，即在模式字符串前加 r 或 R。例如，上面的模式字符串采用原生字符串表示就是：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
r'\bm\w*\b'
```

> **说明：**在编写模式字符串时，并不是所有的反斜杠都需要进行转换，例如，前面编写的正则表达式“^\d{8}$”中的反斜杠就不需要转义，因为其中的\d 并没有特殊意义。不过，为了编写方便，本书中所写正则表达式都采用原生字符串表示。

6.3 使用re模块实现正则表达式操作

在 6.2 节介绍了正则表达式的语法，本节将介绍如何在 Python 中使用正则表达式。Python 提供了 re 模块，用于实现正则表达式的操作，可以使用 re 模块提供的方法（例如，search()、match()、findall()等）进行字符串处理，也可以先使用 re 模块的 compile()方法将模式字符串转换为正则表达式对象，然后再使用该正则表达式对象的相关方法来操作字符串。

在使用 re 模块时，需要先应用 import 语句引入，具体代码如下：

```
import re
```

如果在使用 re 模块时，未将其引入，将抛出如图 6.14 所示的异常。

```
Traceback (most recent call last):
  File "E:\program\Python\Code\test.py", line 22, in <module>
    pattern =re.compile(pattern)
NameError: name 're' is not defined
>>>
```

图 6.14 未引入 re 模块抛出异常

6.3.1 匹配字符串

匹配字符串可以使用 re 模块提供的 match()、search()和 findall()等方法。下面分别进行介绍。

1. 使用 match()方法进行匹配

match()方法用于从字符串的开始处进行匹配，如果在起始位置匹配成功，则返回 Match 对象，否则返回 None，其语法格式如下：

```
re.match(pattern, string, [flags])
```

参数说明如下。

- pattern：表示模式字符串，由要匹配的正则表达式转换而来。
- string：表示要匹配的字符串。
- flags：可选参数，表示标志位，用于控制匹配方式，如是否区分字母大小写。常用的标志如表 6.5 所示。

表 6.5 常用标志

标　志	说　明
A 或 ASCII	对于\w、\W、\b、\B、\d、\D、\s 和\S 只进行 ASCII 匹配（仅适用于 Python 3.x）
I 或 IGNORECASE	执行不区分字母大小写的匹配

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

标　志	说　明
M 或 MULTILINE	将^和$用于包括整个字符串的开始和结尾的每一行（在默认情况下，仅适用于整个字符串的开始和结尾处）
S 或 DOTALL	使用“.”字符匹配所有字符，包括换行符
X 或 VERBOSE	忽略模式字符串中未转义的空格和注释

例如，匹配字符串是否以“mr_”开头，不区分字母大小写，代码如下：

```
01 import re
02 pattern = r'mr_\w+'                      # 模式字符串
03 string = 'MR_SHOP mr_shop'               # 要匹配的字符串
04 match = re.match(pattern,string,re.I)  # 匹配字符串，不区分大小写
05 print(match)                             # 输出匹配结果
06 string = '项目名称 MR_SHOP mr_shop'
07 match = re.match(pattern,string,re.I)  # 匹配字符串，不区分大小写
08 print(match)                             # 输出匹配结果
```

执行结果如下：

```
<_sre.SRE_Match object; span=(0, 7), match='MR_SHOP'>
None
```

从上面的执行结果中可以看出，字符串“MR_SHOP”是以“mr_”开头的，所以返回一个 Match 对象，而字符串“项目名称 MR_SHOP”不是以“mr_”开头的，所以返回“None”。这是因为 match()方法从字符串的开始位置开始匹配，当第一个字母不符合条件时，则不再进行匹配，直接返回 None。

Match 对象中包含了匹配值的位置和匹配数据。其中，要获取匹配值的起始位置可以使用 Match 对象的 start()方法；要获取匹配值的结束位置可以使用 end()方法；通过 span()方法可以返回匹配位置的元组；通过 string 属性可以获取要匹配的字符串。例如下面的代码：

```
01 import re
02 pattern = r'mr_\w+'                      # 模式字符串
03 string = 'MR_SHOP mr_shop'               # 要匹配的字符串
04 match = re.match(pattern,string,re.I)  # 匹配字符串，不区分大小写
05 print('匹配值的起始位置：',match.start())
06 print('匹配值的结束位置：',match.end())
07 print('匹配位置的元组：',match.span())
08 print('要匹配的字符串：',match.string)
09 print('匹配数据：',match.group())
```

执行结果如下：

```
匹配值的起始位置： 0
匹配值的结束位置： 7
匹配位置的元组： (0, 7)
要匹配的字符串： MR_SHOP mr_shop
匹配数据： MR_SHOP
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2．使用 search()方法进行匹配

search()方法用于在整个字符串中搜索第一个匹配的值，如果在起始位置匹配成功，则返回 Match 对象，否则返回 None，其语法格式如下：

```
re.search(pattern, string, [flags])
```

参数说明如下。

- pattern：表示模式字符串，由要匹配的正则表达式转换而来。
- string：表示要匹配的字符串。
- flags：可选参数，表示标志位，用于控制匹配方式，如是否区分字母大小写。常用的标志如表 6.5 所示。

例如，搜索第一个以“mr_”开头的字符串，不区分字母大小写，代码如下：

```
01 import re
02 pattern = r'mr_\w+'                      # 模式字符串
03 string = 'MR_SHOP mr_shop'               # 要匹配的字符串
04 match = re.search(pattern,string,re.I) # 搜索字符串，不区分大小写
05 print(match)                             # 输出匹配结果
06 string = '项目名称 MR_SHOP mr_shop'
07 match = re.search(pattern,string,re.I) # 搜索字符串，不区分大小写
08 print(match)                             # 输出匹配结果
```

执行结果如下：

```
<_sre.SRE_Match object; span=(0, 7), match='MR_SHOP'>
<_sre.SRE_Match object; span=(4, 11), match='MR_SHOP'>
```

从上面的执行结果中可以看出，search()方法不仅仅是在字符串的起始位置搜索，其他位置有符合的匹配也可以。

3．使用 findall()方法进行匹配

findall()方法用于在整个字符串中搜索所有符合正则表达式的字符串，并以列表的形式返回。如果匹配成功，则返回包含匹配结构的列表，否则返回空列表，其语法格式如下：

```
re.findall(pattern, string, [flags])
```

参数说明如下。

- pattern：表示模式字符串，由要匹配的正则表达式转换而来。
- string：表示要匹配的字符串。
- flags：可选参数，表示标志位，用于控制匹配方式，如是否区分字母大小写。常用的标志如表 6.5 所示。

例如，搜索以“mr_”开头的字符串，代码如下：

```
01 import re
02 pattern = r'mr_\w+'                          # 模式字符串
03 string = 'MR_SHOP mr_shop'                   # 要匹配的字符串
04 match = re.findall(pattern,string,re.I)     # 搜索字符串，不区分大小写
05 print(match)                                 # 输出匹配结果
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
06  string = '项目名称 MR_SHOP mr_shop'
07  match = re.findall(pattern,string)          # 搜索字符串，区分大小写
08  print(match)                                # 输出匹配结果
```

执行结果如下：

```
['MR_SHOP', 'mr_shop']
['mr_shop']
```

如果在指定的模式字符串中包含分组，则返回与分组匹配的文本列表，例如：

```
01  import re
02  pattern = r'[1-9]{1,3}(\.[0-9]{1,3}){3}'   # 模式字符串
03  str1 = '127.0.0.1 192.168.1.66'            # 要配置的字符串
04  match = re.findall(pattern,str1)           # 进行模式匹配
05  print(match)
```

上面代码的执行结果如下：

```
['.1', '.66']
```

从上面的结果中可以看出，并没有得到匹配的 IP 地址，这是因为在模式字符串中出现了分组，所以得到的结果是根据分组进行匹配的结果，即“(\.[0–9]{1,3})”匹配的结果。如果想获取整个模式字符串的匹配，可以将整个模式字符串使用一对小括号进行分组，然后在获取结果时只取返回值列表的每个元素（是一个元组）的第 1 个元素，代码如下：

```
01  import re
02  pattern = r'([1-9]{1,3}(\.[0-9]{1,3}){3})' # 模式字符串
03  str1 = '127.0.0.1 192.168.1.66'            # 要配置的字符串
04  match = re.findall(pattern,str1)           # 进行模式匹配
05  for item in match:
06      print(item[0])
```

执行结果如下：

```
127.0.0.1
192.168.1.66
```

6.3.2　替换字符串

sub()方法用于实现字符串替换，其语法格式如下：

```
re.sub(pattern, repl, string, count, flags)
```

参数说明如下。

- pattern：表示模式字符串，由要匹配的正则表达式转换而来。
- repl：表示替换的字符串。
- string：表示要被查找替换的原始字符串。
- count：可选参数，表示模式匹配后替换的最大次数，默认值为 0，表示替换所有的匹配。
- flags：可选参数，表示标志位，用于控制匹配方式，如是否区分字母大小写。常用的标志如表 6.5 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

例如，隐藏中奖信息中的手机号码，代码如下：

```
01 import re
02 pattern = r'1[34578]\d{9}'                          # 定义要替换的模式字符串
03 string = '中奖号码为：84978981 联系电话为：13611111111'
04 result = re.sub(pattern,'1XXXXXXXXXX',string)  # 替换字符串
05 print(result)
```

执行结果如下：

```
中奖号码为：84978981 联系电话为：1XXXXXXXXXX
```

6.3.3 使用正则表达式分隔字符串

split()方法用于实现根据正则表达式分隔字符串，并以列表的形式返回。其作用同 6.1.4 节介绍的字符串对象的 split()方法类似，所不同的就是分隔字符由模式字符串指定。其语法格式如下：

```
re.split(pattern, string, [maxsplit], [flags])
```

参数说明如下。

- pattern：表示模式字符串，由要匹配的正则表达式转换而来。
- string：表示要匹配的字符串。
- maxsplit：可选参数，表示最大的拆分次数。
- flags：可选参数，表示标志位，用于控制匹配方式，如是否区分字母大小写。常用的标志如表 6.5 所示。

例如，从给定的 URL 地址中提取出请求地址和各个参数，代码如下：

```
01 import re
02 pattern = r'[?|&]'                    # 定义分隔符
03 url = 'http://www.mingrisoft.com/login.jsp?username="mr"&pwd="mrsoft"'
04 result = re.split(pattern,url)      # 分隔字符串
05 print(result)
```

执行结果如下：

```
['http://www.mingrisoft.com/login.jsp', 'username="mr"', 'pwd="mrsoft"']
```

6.4 快学快用

（1）已知字符串 word="aS98hdkHFSWEas754"，编写一个程序，分别实现如下功能：

- 请将 word 字符串中的大写字母改为小写字母，小写字母改为大写字母。
- 请将 word 字符串中的数字取出，并输出成一个新的字符串。

（2）已知字符串 word="ABDPISDNGDA"，请去除 word 字符串多次出现的字母，仅留最先出现的一个。例如 aabc，经过去除后，输出 abc。

（3）张三的身份证号码是“210011196809210451”，请提取张三的生日信息并输出。

電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（4）将一句英文句子单词顺序倒序输出，但是不改变单词结构。例如：I am a beautiful girl，输出为 girl beautiful a am I。

（5）Python 提供了很多有用的内置函数，大家熟悉一下这些函数的用途，在实际开发时如果遇到相关应用，可以联想到相关函数，然后详细学习并使用，事半功倍。集合类操作内置函数如表 6.6 所示。如果要详细了解这些函数的使用，可以扫描二维码，到快用云端支持平台学习、训练、实战。

表 6.6　集合类操作内置函数（1）

basestring()	str 和 unicode 的超类 不能直接调用，可以用作 isinstance 判断
format(value [, format_spec])	格式化输出字符串 格式化的参数顺序从 0 开始，如“I am {0},I like {1}”
unichr(i)	返回给定 int 类型的 unicode
enumerate(sequence [, start = 0])	返回一个可枚举的对象，该对象的 next()方法将返回一个 tuple
iter(o[, sentinel])	生成一个对象的迭代器，第二个参数表示分隔符
max(iterable[, args...][key])	返回集合中的最大值
min(iterable[, args...][key])	返回集合中的最小值

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 7 章

异常处理及程序调试

学习过 C 语言或者 Java 语言的用户都知道，在 C 语言或者 Java 语言中，编译器可以捕获很多语法错误。但是，在 Python 语言中，只有在程序运行后才会执行语法检查，所以，只有在运行或测试程序时，才会真正知道该程序能不能正常运行。因此，掌握一定的异常处理语句和程序调试方法是十分必要的。

本章将主要介绍常用的异常处理语句，以及如何使用 assert 语句进行调试。

7.1 异常概述

在程序运行过程中，经常会遇到各种各样的错误，这些错误统称为“异常”。这些异常有的是由于开发者一时疏忽将关键字敲错导致的，这类错误多数产生的是 SyntaxError: invalid syntax（无效的语法），这将直接导致程序不能运行。这类异常是显式的，在开发阶段很容易发现。还有一类是隐式的，通常和使用者的操作有关。

实例 01　异常

场景模拟：在全民学编程的时代，作为程序员二代的小琦编写了一个程序，模拟幼儿园老师分苹果。如果老师买来 10 个苹果，今天来了 10 个小朋友，那么输入 10 和 10，程序给出的结果是每人分 1 个苹果。但是小琦的程序有一个异常，下面来看一下。

在 IDLE 中创建一个名称为 division_apple.py 的文件，然后在该文件中定义一个模拟分苹果的函数 division()，在该函数中，要求输入苹果的数量和小朋友的数量，然后应用除法算式计算分配的结果，最后调用 division()函数，代码如下：

```
01 def division():
02     '''功能：分苹果'''
03     print("\n===================== 分苹果了 =====================\n")
04     apple = int(input("请输入苹果的个数："))          # 输入苹果的个数
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
05     children = int(input("请输入来了几个小朋友: "))
06     result = apple//children                          # 计算每人分几个苹果
07     remain =apple-result*children                     # 计算余下几个苹果
08     if remain>0:
09        print(apple,"个苹果，平均分给",children,"个小朋友，每人分",result,
10              "个,剩下",remain,"个。")
11     else:
12        print(apple,"个苹果，平均分给",children,"个小朋友，每人分",result,"个。")
13 if __name__ == '__main__':
14     division()                                        # 调用分苹果的函数
```

运行程序，当输入苹果和小朋友的个数都是 10 时，将显示如图 7.1 所示的结果。

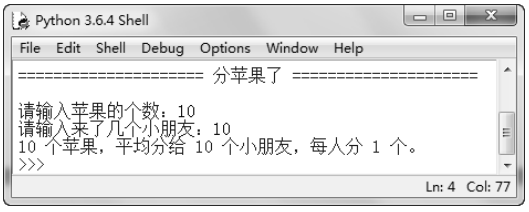


图 7.1　正确的输出结果

如果在输入个数时，不小心把小朋友的个数输成了 0，将得到如图 7.2 所示的结果。

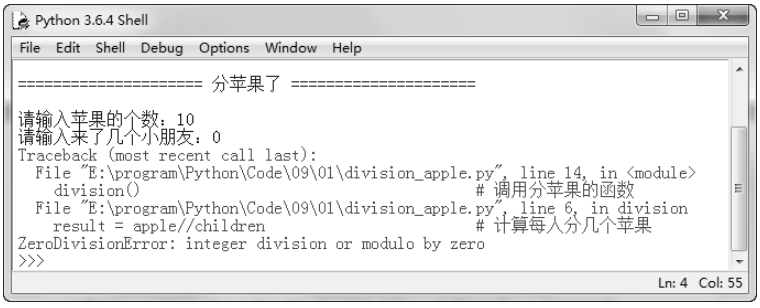


图 7.2　抛出了 ZeroDivisionError 异常

产生 ZeroDivisionError（除数为 0 错误）的根源在于算术表达式“10/0”中，0 作为除数出现，所以正在执行的程序被中断（第 6 行以后，包括第 6 行的代码都不会被执行）。

除了 ZeroDivisionError 异常外，Python 中还有很多异常。如表 7.1 所示为 Python 中常见的异常。

表 7.1　Python 中常见的异常

异　常	描　述
NameError	尝试访问一个没有声明的变量引发的错误
IndexError	索引超出序列范围引发的错误
IndentationError	缩进错误
ValueError	传入的值错误
KeyError	请求一个不存在的字典关键字引发的错误

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

异　常	描　述
IOError	输入/输出错误（如要读取的文件不存在）
ImportError	当 import 语句无法找到模块或 from 无法在模块中找到相应的名称时引发的错误
AttributeError	尝试访问未知的对象属性引发的错误
TypeError	类型不合适引发的错误
MemoryError	内存不足引发的错误
ZeroDivisionError	除数为 0 引发的错误

说明：表 7.1 所示的异常并不需要记住，只要简单了解即可。

7.2　异常处理语句

在程序开发时，有些错误并不是每次运行都会出现。例如，本章的实例 01，只要输入的数据符合程序的要求，程序就可以正常运行，否则将抛出异常并停止运行。假设在输入苹果的数量时，输入了 23.5，那么程序将抛出如图 7.3 所示的异常。

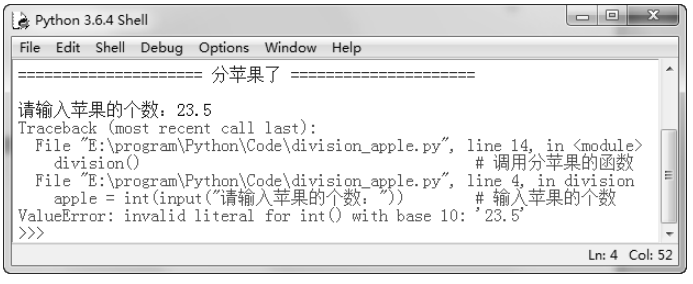


图 7.3　抛出 ValueError 异常

这时，就需要在开发程序时对可以出现异常的情况进行处理。下面将详细介绍 Python 中提供的异常处理语句。

7.2.1　try…except语句

在 Python 中，提供了 try…except 语句捕获并处理异常。在使用时，把可能产生异常的代码放在 try 语句块中，把处理结果放在 except 语句块中，这样，当 try 语句块中的代码出现错误时，就会执行 except 语句块中的代码，如果 try 语句块中的代码没有错误，那么 except 语句块将不会执行。具体的语法格式如下：

```
try:
    block1
except [ExceptionName [as alias]]:
    block2
```

参数说明如下。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- block1：表示可能出现错误的代码块。
- ExceptionName [as alias]：可选参数，用于指定要捕获的异常。其中，ExceptionName 表示要捕获的异常名称，如果在其右侧加上 as alias 则表示为当前的异常指定一个别名，通过该别名，可以记录异常的具体内容。

> **说明：**在使用 try…except 语句捕获异常时，如果在 except 后面不指定异常名称，则表示捕获全部异常。

- block2：表示进行异常处理的代码块。在这里可以输出固定的提示信息，也可以通过别名输出异常的具体内容。

> **说明：**使用 try…except 语句捕获异常后，当程序出错时，输出错误信息后，程序会继续执行。

7.2.2　try…except…else语句

在 Python 中，还有另一种异常处理结构，它是 try…except…else 语句，也就是在原来 try…except 语句的基础上再添加一个 else 子句，用于指定当 try 语句块中没有发现异常时要执行的语句块。当 try 语句中发现异常时，该语句块中的内容将不被执行。例如，对实例 01 进行修改，实现当 division()函数在执行时没有抛出异常时，输出文字“分苹果顺利完成…”，修改后的代码如下：

```
def division():
    '''功能：分苹果'''
    print("\n====================== 分苹果了 ======================\n")
    apple = int(input("请输入苹果的个数："))           # 输入苹果的个数
    children = int(input("请输入来了几个小朋友："))
    result = apple // children                        # 计算每人分几个苹果
    remain = apple - result * children                # 计算余下几个苹果
    if remain > 0:
        print(apple, "个苹果，平均分给", children, "个小朋友，每人分", result,
              "个,剩下", remain, "个。")
    else:
        print(apple, "个苹果，平均分给", children, "个小朋友，每人分", result, "个。")
if __name__ == '__main__':
    try:                                              # 捕获异常
        division()                                    # 调用分苹果的函数
    except ZeroDivisionError:                         # 处理异常
        print("\n出错了 ~_~ ——苹果不能被 0 个小朋友分！")
    except ValueError as e:                           # 处理 ValueError 异常
        print("输入错误：", e)                         # 输出错误原因
    else:                                             # 没有抛出异常时执行
        print("分苹果顺利完成...")
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

执行代码，将显示如图 7.4 所示的运行结果。

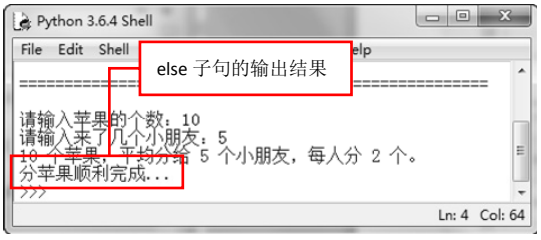


图 7.4　没有抛出异常时提示相应信息

7.2.3　try…except…finally语句

完整的异常处理语句应该包含 finally 代码块，在通常情况下，无论程序中有无异常产生，finally 代码块中的代码都会被执行。其基本格式如下：

```
try:
    block1
except [ExceptionName [as alias]]:
    block2
finally:
    block3
```

对于 try…except…finally 语句的理解并不复杂，它只是比 try…except 语句多了一个 finally 语句，如果程序中有一些在任何情形中都必须执行的代码，那么就可以将它们放在 finally 语句的代码块中。

> **说明：**使用 except 子句是为了允许处理异常。无论是否引发了异常，使用 finally 子句都可以执行清理代码。如果分配了有限的资源（如打开文件），则应将释放这些资源的代码放置在 finally 代码块中。

例如，再对实例 01 进行修改，实现当 division()函数在执行时无论是否抛出异常，都输出文字“进行了一次分苹果操作。”。修改后的代码如下：

```
01 def division():
02     '''功能：分苹果'''
03     print("\n====================== 分苹果了 ======================\n")
04     apple = int(input("请输入苹果的个数："))                # 输入苹果的个数
05     children = int(input("请输入来了几个小朋友："))
06     result = apple // children                              # 计算每人分几个苹果
07     remain = apple - result * children                      # 计算余下几个苹果
08     if remain > 0:
09         print(apple, "个苹果，平均分给", children, "个小朋友，每人分", result,
10               "个，剩下", remain, "个。")
11     else:
12        print(apple, "个苹果，平均分给", children, "个小朋友，每人分", result, "个。")
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
13 if __name__ == '__main__':
14     try:                                                    # 捕获异常
15         division()                                          # 调用分苹果的函数
16     except ZeroDivisionError:                               # 处理异常
17         print("\n出错了 ~_~ ——苹果不能被 0 个小朋友分! ")
18     except ValueError as e:                                 # 处理 ValueError 异常
19         print("输入错误: ", e)                               # 输出错误原因
20     else:                                                   # 没有抛出异常时执行
21         print("分苹果顺利完成...")
22     finally:                                                # 无论是否抛出异常都执行
23         print("进行了一次分苹果操作。")
```

执行代码，将显示如图 7.5 所示的运行结果。

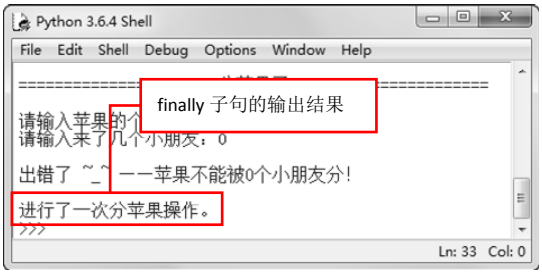


图 7.5　没有抛出异常时提示相应信息

至此，已经介绍了异常处理语句的 try…except、try…except…else 和 try…except…finally 等形式。下面通过图 7.6 说明异常处理语句的各个子句的执行关系。

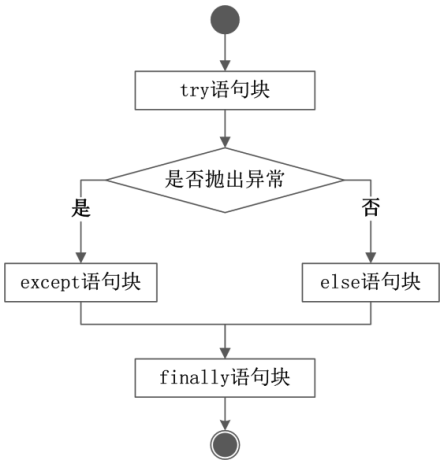


图 7.6　异常处理语句的不同子句的执行关系

7.2.4　使用raise语句抛出异常

如果某个函数或方法可能会产生异常，但不想在当前函数或方法中处理这个异常，则可以使用 raise 语句在函数或方法中抛出异常。raise 语句的基本格式如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
raise [ExceptionName[(reason)]]
```

其中，ExceptionName[(reason)]为可选参数，用于指定抛出的异常名称，以及异常信息的相关描述。如果省略，就会把当前的错误原样抛出。

> **说明：**ExceptionName(reason)参数中的(reason)也可以省略，如果省略，则在抛出异常时不附带任何描述信息。

7.3 使用assert语句调试程序

在程序开发过程中，免不了会出现一些错误，有语法方面的，也有逻辑方面的。对于语法方面的错误比较好检测，因为程序会直接停止，并且给出错误提示。而对于逻辑错误就不太容易发现了，因为程序可能会一直执行下去，但结果是错误的，所以作为一名程序员，掌握一定的程序调试方法，可以说是一项必备技能。

Python 提供了 assert 语句来调试程序。assert 的中文意思是断言，它一般用于对程序某个时刻必须满足的条件进行验证。assert 语句的基本语法如下：

```
assert expression [,reason]
```

参数说明如下。

- expression：条件表达式，如果该表达式的值为真，什么都不做；如果为假，则抛出 AssertionError 异常。
- reason：可选参数，用于对判断条件进行描述，为了以后更好地知道哪里出现了问题。

例如，修改实例 01，加入限制苹果个数必须大于或等于小朋友的个数，从而保证每个小朋友都能至少得到一个苹果。

在 IDLE 中创建一个名称为 division_apple_dug.py 的文件，然后将实例 01 的代码全部复制到该文件中，并且在第 5 行代码“children = int(input("请输入来了几个小朋友："))”的下方添加一个 assert 语句，验证苹果的个数是否小于小朋友的个数，修改后的代码如下：

```
01 def division():
02     '''功能：分苹果'''
03     print("\n====================== 分苹果了 ======================\n")
04     apple = int(input("请输入苹果的个数："))           # 输入苹果的个数
05     children = int(input("请输入来了几个小朋友："))
06     assert apple > children ,"苹果不够分"              # 应用断言调试
07     result = apple // children                        # 计算每人分几个苹果
08     remain = apple - result * children                # 计算余下几个苹果
09     if remain > 0:
10         print(apple, "个苹果，平均分给", children, "个小朋友，每人分", result,
11               "个，剩下", remain, "个。")
12     else:
13        print(apple, "个苹果，平均分给", children, "个小朋友，每人分", result, "个。")
```


電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
14 if __name__ == '__main__':
15     division()                                    # 调用分苹果的函数
```

运行程序，输入苹果的个数为 5，小朋友的个数为 10 时，将抛出如图 7.7 所示的 AssertionError 异常。

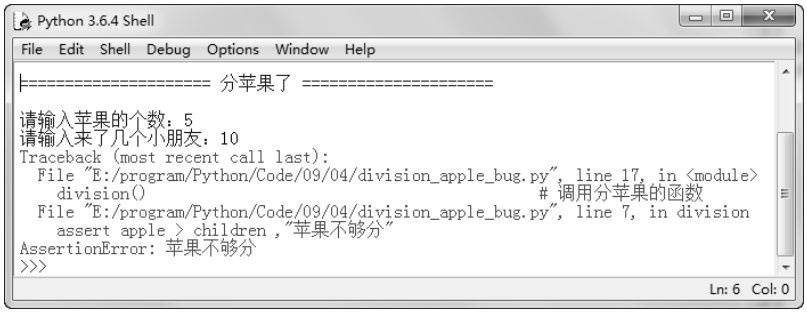


图 7.7　苹果的个数小于小朋友的个数时抛出 AssertionError 异常

在通常情况下，assert 语句可以和异常处理语句结合使用。所以，可以将上面代码的第 15 行修改为以下内容：

```
01 try:
02     division()                                    # 调用分苹果的函数
03 except AssertionError as e:                       # 处理 AssertionError 异常
04     print("\n 输入有误：",e)
```

这样，再执行程序就不会直接抛出异常，而是给出如图 7.8 所示的友好提示。

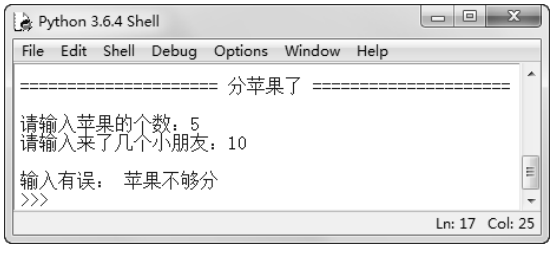


图 7.8　处理抛出的 AssertionError 异常

assert 语句只在调试阶段有效。我们可以通过在执行 python 命令时加入-O（大写字母）参数来关闭 assert 语句。例如，在命令行窗口中输入以下代码执行 E:\program\Python\Code 目录下的 division_apple_bug.py 文件，即关闭 division_apple_bug.py 文件中的 assert 语句。

```
01 E:
02 cd E:\program\Python\Code
03 python -O division_apple_bug.py
```

执行上面的语句后，输入苹果的个数为 5，小朋友的个数为 10 时，并没有给出“输入有误：苹果不够分”的提示，如图 7.9 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

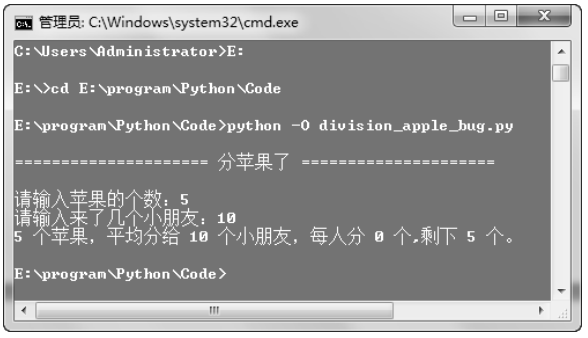


图 7.9　在非调试状态下执行程序将忽略 assert 语句

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 8 章

函 数

在前面的章节中，所有编写的代码都是从上到下依次执行的，如果某段代码需要多次使用，则需要将该段代码多次复制，这种做法势必会影响开发效率，在实际项目开发中是不可取的。那么如果想要多次使用某段代码，应该怎么做呢？在 Python 中，可以使用函数来达到这个目的。我们把实现某一功能的代码定义为一个函数，在需要使用时，随时调用即可，十分方便。对于函数，简单理解就是可以完成某项工作的代码块，类似于积木块，可以反复使用。

本章将对如何创建和调用函数、函数的参数、变量的作用域等进行详细介绍。

8.1 函数的创建和调用

提到函数，大家会想到数学函数吧，函数是数学最重要的一个模块，贯穿整个数学领域。在 Python 中，函数的应用也非常广泛。在前面我们已经多次接触过函数。例如，用于输出的 print()函数、用于输入的 input()函数，以及用于生成一系列整数的 range()函数，这些都是 Python 内置的标准函数，可以直接使用。除了可以直接使用的标准函数外，Python 还支持自定义函数，即通过将一段有规律的、重复的代码定义为函数，来达到一次编写多次调用的目的。使用函数可以提高代码的重复利用率。

8.1.1 创建函数

创建函数也称为定义函数，可以理解为创建一个具有某种用途的工具。使用 def 关键字实现创建函数，具体的语法格式如下：

```
def functionname([parameterlist]):
    ['''comments''']
    [functionbody]
```

参数说明如下。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- functionname：函数名称，在调用函数时使用。
- parameterlist：可选参数，用于指定向函数中传递的参数。如果有多个参数，各参数间使用逗号“,”进行分隔。如果不指定，则表示该函数没有参数，在调用时，也不指定参数。

> **注意：**即使函数没有参数，也必须保留一对空的小括号“()”，否则将显示如图 8.1 所示的语法错误提示。

图 8.1　语法错误提示

- '"comments"'：可选参数，表示为函数指定注释，注释的内容通常是说明该函数的功能、要传递的参数的作用等，可以为用户提供友好提示和帮助的内容。

> **说明：**在定义函数时，如果指定了'"comments"'参数，那么在调用函数时，输入函数名称及左侧的小括号时，就会显示该函数的帮助信息，如图 8.2 所示。这些帮助信息就是通过定义的注释提供的。

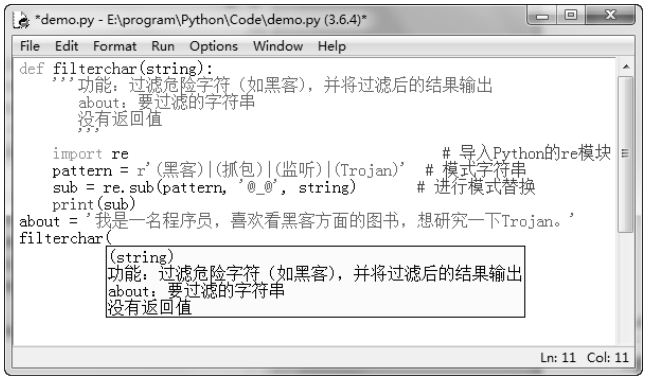


图 8.2　调用函数时显示友好提示

> **注意：**如果在输入函数名和左侧括号后，没有显示友好提示，那么就检查函数本身是否有误，检查方法可以是在未调用该函数时，先按下快捷键<F5>执行一遍代码。

- functionbody：可选参数，用于指定函数体，即该函数被调用后，要执行的功能代码。如果函数有返回值，可以使用 return 语句返回。

> **注意：**函数体“functionbody”和注释“'"comments"'”相对于 def 关键字必须保持一定的缩进。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

说明：如果想定义一个什么也不做的空函数，可以使用 pass 语句作为占位符。

例如，定义一个过滤危险字符的函数 filterchar()，代码如下：

```
def filterchar(string):
    '''功能：过滤危险字符（如黑客），并将过滤后的结果输出
       about：要过滤的字符串
       没有返回值
       '''
    import re                                          # 导入 Python 的 re 模块
    pattern = r'(黑客)|(抓包)|(监听)|(Trojan)'          # 模式字符串
    sub = re.sub(pattern, '@_@', string)               # 进行模式替换
    print(sub)
```

运行上面的代码，将不显示任何内容，也不会抛出异常，因为 filterchar()函数还没有调用。

8.1.2　调用函数

调用函数也就是执行函数。如果把创建函数理解为创建一个具有某种用途的工具，那么调用函数就相当于使用该工具。调用函数的基本语法格式如下：

```
functionname([parametersvalue])
```

参数说明如下。

- functionname：函数名称，要调用的函数名称，必须是已经创建好的。
- parametersvalue：可选参数，用于指定各个参数的值。如果需要传递多个参数值，则各参数值间使用逗号“,”进行分隔。如果该函数没有参数，则直接写一对小括号即可。

例如，调用在 8.1.1 节创建的 filterchar()函数，可以使用下面的代码：

```
about = '我是一名程序员，喜欢看@_@方面的图书，想研究一下@_@。'
filterchar(about)
```

调用 filterchar()函数后，将显示如图 8.3 所示的结果。

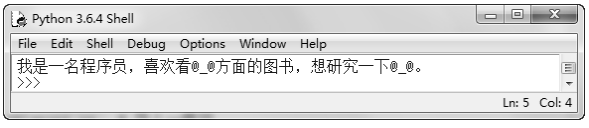


图 8.3　调用 filterchar()函数的结果

8.2　参数传递

在调用函数时，在大多数情况下，主调用函数和被调用函数之间有数据传递关系，这就是有参数的函数形式。函数参数的作用是传递数据给函数使用，函数利用接收的数据进行具体的操作处理。

函数参数在定义函数时放在函数名称后面的一对小括号中，如图 8.4 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

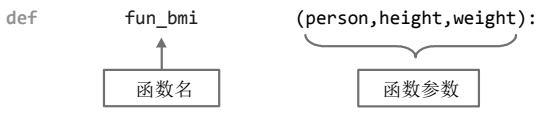


图 8.4　函数参数

8.2.1　了解形式参数和实际参数

在使用函数时，经常会用到形式参数和实际参数，两者都叫作参数，两者之间的区别将先通过形式参数与实际参数的作用来进行讲解，再通过一个比喻进行深入理解。

1．通过作用理解

形式参数和实际参数在作用上的区别如下。

- 形式参数：在定义函数时，函数名后面括号中的参数为“形式参数”。
- 实际参数：在调用一个函数时，函数名后面括号中的参数为“实际参数”，也就是将函数的调用者提供给函数的参数称为实际参数。通过图 8.5 可以更好地理解。

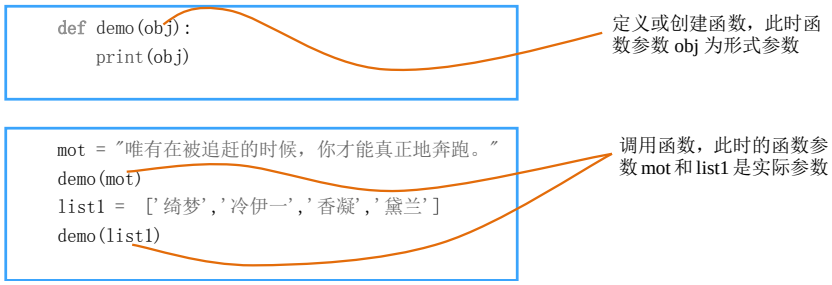


图 8.5　形式参数与实际参数

根据实际参数的类型不同，可以分为将实际参数的值传递给形式参数，将实际参数的引用传递给形式参数两种情况。其中，当实际参数为不可变对象时，进行的是值传递；当实际参数为可变对象时，进行的是引用传递。实际上，值传递和引用传递的基本区别就是，进行值传递后，改变形式参数的值，实际参数的值不变；而进行引用传递后，改变形式参数的值，实际参数的值也一同改变。

例如，定义一个名称为 demo 的函数，然后为 demo()函数传递一个字符串类型的变量作为参数（代表值传递），并在函数调用前后分别输出该字符串变量，再为 demo()函数传递一个列表类型的变量作为参数（代表引用传递），并在函数调用前后分别输出该列表。代码如下：

```
# 定义函数
def demo(obj):
    print("原值：",obj)
    obj += obj
#调用函数
print("=========值传递========")
mot = "唯有在被追赶的时候，你才能真正地奔跑。"
print("函数调用前：",mot)
```


電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
09 demo(mot)  #采用不可变对象——字符串
10 print("函数调用后: ",mot)
11 print("=========引用传递 ========")
12 list1 =  ['绮梦','冷伊一','香凝','黛兰']
13 print("函数调用前: ",list1)
14 demo(list1)  #采用可变对象——列表
15 print("函数调用后: ",list1)
```

上面代码的执行结果如下：

```
=========值传递========
函数调用前:  唯有在被追赶的时候，你才能真正地奔跑。
原值:  唯有在被追赶的时候，你才能真正地奔跑。
函数调用后:  唯有在被追赶的时候，你才能真正地奔跑。
=========引用传递 ========
函数调用前:  ['绮梦', '冷伊一', '香凝', '黛兰']
原值:  ['绮梦', '冷伊一', '香凝', '黛兰']
函数调用后:  ['绮梦', '冷伊一', '香凝', '黛兰', '绮梦', '冷伊一', '香凝', '黛兰']
```

从上面的执行结果中可以看出，在进行值传递时，改变形式参数的值后，实际参数的值不改变；在进行引用传递时，改变形式参数的值后，实际参数的值也发生改变。

2．通过一个比喻来理解形式参数和实际参数

函数定义时参数列表中的参数就是形式参数，而函数调用时传递进来的参数就是实际参数。就像剧本选主角一样，剧本的角色相当于形式参数，而演角色的演员就相当于实际参数。

8.2.2　位置参数

位置参数也称为必备参数，必须按照正确的顺序传到函数中，即调用时的数量和位置必须和定义时是一样的。下面分别进行介绍。

1．数量必须与定义时一致

在调用函数时，指定的实际参数的数量必须与形式参数的数量一致，否则将抛出 TypeError 异常，提示缺少必要的位置参数。

例如，调用根据身高、体重计算 BMI 指数的函数 fun_bmi(person,height,weight)，将参数少传一个，即只传递两个参数，代码如下：

```
fun_bmi("路人甲",1.83)       # 计算路人甲的 BMI 指数
```

函数调用后，将显示如图 8.6 所示的异常信息。

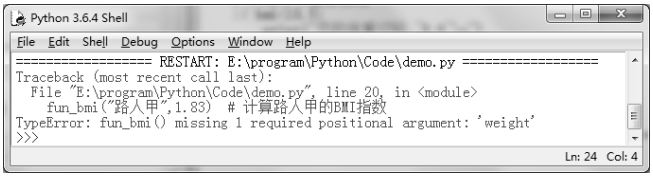


图 8.6　缺少必要的参数抛出的异常

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

从图 8.6 所示的异常信息中可以看出，抛出的异常类型为 TypeError，具体的意思是“fun_bmi()方法缺少一个必要的位置参数 weight”。

2．位置必须与定义时一致

在调用函数时，指定的实际参数的位置必须与形式参数的位置一致，否则将产生以下两种结果。

（1）抛出 TypeError 异常

抛出异常的情况主要是因为实际参数的类型与形式参数的类型不一致，并且在函数中，这两种类型还不能正常转换。

例如，调用 fun_bmi(person,height,weight)函数，将第 1 个参数和第 2 个参数位置调换，代码如下：

```
fun_bmi(60, "路人甲",1.83)        # 计算路人甲的 BMI 指数
```

函数调用后，将显示如图 8.7 所示的异常信息。主要是因为传递的整型数值不能与字符串进行连接操作。

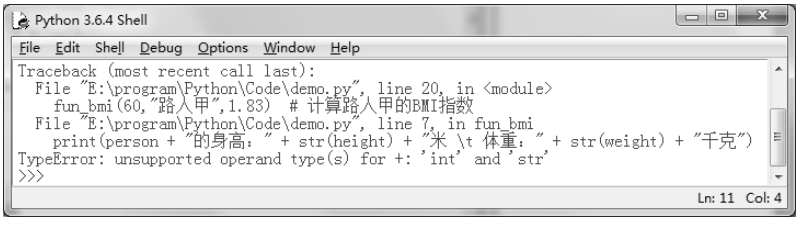


图 8.7　提示不支持的操作数类型

（2）产生的结果与预期不符

在调用函数时，如果指定的实际参数与形式参数的位置不一致，但是它们的数据类型一致，那么就不会抛出异常，但是产生的结果与预期不符。

例如，调用 fun_bmi(person,height,weight)函数，将第 2 个参数和第 3 个参数位置调换，代码如下：

```
fun_bmi("路人甲",60,1.83)        # 计算路人甲的 BMI 指数
```

函数调用后，将显示如图 8.8 所示的结果。从结果中可以看出，虽然没有抛出异常，但是得到的结果与预期不符。

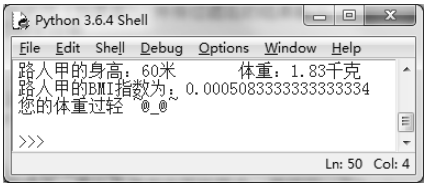


图 8.8　结果与预期不符

> **说明：**由于调用函数时，传递的实际参数的位置与形式参数的位置不一致时，并不会总是抛出异常，所以在调用函数时一定要确定好位置，否则产生 Bug，还不容易被发现。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

8.2.3　关键字参数

关键字参数是指使用形式参数的名字来确定输入的参数值，通过该方式指定实际参数时，不再需要与形式参数的位置完全一致，只要将参数名写正确即可，这样可以避免用户需要牢记参数位置的麻烦，使得函数的调用和参数传递更加灵活方便。

例如，调用 fun_bmi(person,height,weight)函数，通过关键字参数指定各个实际参数，代码如下：

```
fun_bmi( height = 1.83, weight = 60, person = "路人甲")   # 计算路人甲的 BMI 指数
```

函数调用后，将显示以下结果：

```
路人甲的身高：1.83 米　体重：60 千克
路人甲的 BMI 指数为：17.916330735465376
您的体重过轻 ~@_@~
```

从上面的结果中可以看出，虽然在指定实际参数时，顺序与定义函数时不一致，但是运行结果与预期是一致的。

8.2.4　为参数设置默认值

在调用函数时，如果没有指定某个参数将抛出异常，即在定义函数时，直接指定形式参数的默认值。这样，当没有传入参数时，则直接使用定义函数时设置的默认值。定义带有默认值参数的函数的语法格式如下：

```
def functionname(…,[parameter1 = defaultvalue1]):
    [functionbody]
```

参数说明如下。

- functionname：函数名称，在调用函数时使用。
- parameter1 = defaultvalue1：可选参数，用于指定向函数中传递的参数，并且为该参数设置默认值为 defaultvalue1。
- functionbody：可选参数，用于指定函数体，即该函数被调用后，要执行的功能代码。

> **注意：**在定义函数时，指定默认的形式参数必须在所有参数的最后，否则将产生语法错误。

多学两招

在 Python 中，可以使用“函数名.__defaults__”查看函数的默认值参数的当前值，其结果是一个元组。例如，显示上面定义的 fun_bmi()函数的默认值参数的当前值，可以使用“fun_bmi.__defaults__”，结果为“('路人',)”。

另外，使用可变对象作为函数参数的默认值时，多次调用可能会导致意料之外的情况。例如，编写一个名称为 demo()的函数，并为其设置一个带默认值的参数，代码如下：

```
01 def demo(obj=[]):  # 定义函数并为参数 obj 指定默认值
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
02     print("obj 的值: ",obj)
03     obj.append(1)
```

调用 demo()函数，代码如下：

```
demo()   #调用函数
```

将显示以下结果：

```
obj 的值: []
```

连续两次调用 demo()函数，并且都不指定实际参数，代码如下：

```
01 demo()   #调用函数
02 demo()   #调用函数
```

将显示以下结果：

```
obj 的值: []
obj 的值: [1]
```

从上面的结果中可以看出，这显然不是我们想要的结果。为了防止出现这种情况，最好使用 None 作为可变对象的默认值，这时还需要加上必要的检查代码。修改后的代码如下：

```
01 def demo(obj=None):
02     if obj==None:
03         obj = []
04     print("obj 的值: ",obj)
05     obj.append(1)
```

将显示以下运行结果：

```
obj 的值: []
obj 的值: []
```

> **说明：**在定义函数时，为形式参数设置默认值要牢记一点：默认参数必须指向不可变对象。

8.2.5 可变参数

在 Python 中，还可以定义可变参数。可变参数也称为不定长参数，即传入函数中的实际参数可以是零个、一个、两个到任意个。

在定义可变参数时，主要有两种形式，一种是*parameter，另一种是**parameter。下面分别进行介绍。

1．*parameter

这种形式表示接收任意多个实际参数并将其放到一个元组中。例如，定义一个函数，让其可以接收任意多个实际参数，代码如下：

```
01 def printcoffee(*coffeename):  # 定义输出我喜欢的咖啡名称的函数
02     print('\n我喜欢的咖啡有: ')
03     for item in coffeename:
04         print(item)  # 输出咖啡名称
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

调用 3 次上面的函数，分别指定不同个数的实际参数，代码如下：

```
01 printcoffee('蓝山')
02 printcoffee('蓝山', '卡布奇诺', '土耳其', '巴西', '哥伦比亚')
03 printcoffee('蓝山', '卡布奇诺', '曼特宁', '摩卡')
```

执行结果如图 8.9 所示。

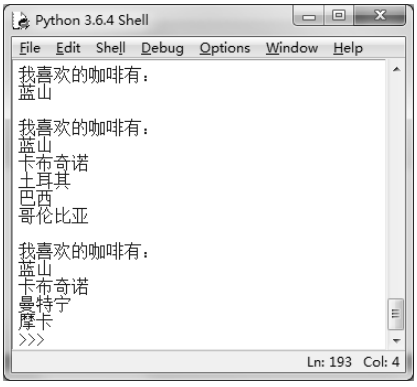


图 8.9　让函数具有可变参数

如果想要使用一个已经存在的列表作为函数的可变参数，可以在列表的名称前加“*”。例如下面的代码：

```
01 param = ['蓝山', '卡布奇诺', '土耳其']     # 定义一个列表
02 printcoffee(*param)                        # 通过列表指定函数的可变参数
```

通过上面的代码调用 printcoffee()函数后，将显示以下运行结果：

```
我喜欢的咖啡有：
蓝山
卡布奇诺
土耳其
```

2．**parameter

这种形式表示接收任意多个类似关键字参数一样显式赋值的实际参数，并将其放到一个字典中。例如，定义一个函数，让其可以接收任意多个显式赋值的实际参数，代码如下：

```
01 def printsign(**sign):                                  # 定义输出姓名和星座的函数
02     print()                                             # 输出一个空行
03     for key, value in sign.items():                     # 遍历字典
04         print("[" + key + "] 的星座是：" + value) # 输出组合后的信息
```

调用两次 printsign()函数，代码如下：

```
01 printsign(绮梦='水瓶座', 冷伊一='射手座')
02 printsign(香凝='双鱼座', 黛兰='双子座', 冷伊一='射手座')
```

执行结果如下：

```
[绮梦] 的星座是：水瓶座
[冷伊一] 的星座是：射手座
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
[香凝] 的星座是：双鱼座
[黛兰] 的星座是：双子座
[冷伊一] 的星座是：射手座
```

如果想要使用一个已经存在的字典作为函数的可变参数，可以在字典的名称前加“**”。例如下面的代码：

```
01  dict1 = {'绮梦': '水瓶座', '冷伊一': '射手座','香凝':'双鱼座'}      # 定义一个字典
02  printsign(**dict1)                                          # 通过字典指定函数的可变参数
```

通过上面的代码调用 printsign()函数后，将显示以下运行结果：

```
[绮梦] 的星座是：水瓶座
[冷伊一] 的星座是：射手座
[香凝] 的星座是：双鱼座
```

8.3 返回值

到目前为止，我们创建的函数都只是为我们做一些事，做完了就结束。但实际上，有时还需要对事情的结果进行获取，这类似于主管向下级职员下达命令，职员去做，最后需要将结果报告给主管。为函数设置返回值的作用就是将函数的处理结果返回给调用它的程序。

在 Python 中，可以在函数体内使用 return 语句为函数指定返回值。该返回值可以是任意类型，并且无论 return 语句出现在函数的什么位置，只要得到执行，就会直接结束函数的执行。

return 语句的语法格式如下：

```
result = return [value]
```

参数说明如下。

- result：用于保存返回结果，如果返回一个值，那么 result 中保存的就是返回的一个值，该值可以是任意类型。如果返回多个值，那么 result 中保存的是一个元组。
- value：可选参数，用于指定要返回的值，可以返回一个值，也可以返回多个值。

> **说明：**当函数中没有 return 语句时，或者省略了 return 语句的参数时，将返回 None，即返回空值。

8.4 变量的作用域

变量的作用域是指程序代码能够访问该变量的区域，如果超出该区域，再访问时就会出现错误。在程序中，一般会根据变量的有效范围将变量分为全局变量和局部变量。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

8.4.1　局部变量

局部变量是指在函数内部定义并使用的变量，它只在函数内部有效，即函数内部的名字只在函数运行时才会创建，在函数运行之前或者运行完毕之后，所有的名字就都不存在了。所以，如果在函数外部使用函数内部定义的变量，就会抛出 NameError 异常。

例如，定义一个名称为 f_demo 的函数，在该函数内部定义一个变量 message（称为局部变量），并为其赋值，然后输出该变量，最后在函数体外部再次输出 message 变量，代码如下：

```
01 def f_demo():
02     message = '唯有在被追赶的时候，你才能真正地奔跑。'
03     print('局部变量 message =',message)       # 输出局部变量的值
04 f_demo()                                      # 调用函数
05 print('局部变量 message =',message)           # 在函数体外输出局部变量的值
```

运行上面的代码将显示如图 8.10 所示的异常。

图 8.10　要访问的变量不存在

8.4.2　全局变量

与局部变量对应，全局变量为能够作用于函数内外的变量。全局变量主要有以下两种情况。

（1）如果一个变量在函数体外定义，那么不仅在函数外可以访问到，在函数内也可以访问到。在函数体以外定义的变量是全局变量。

例如，定义一个全局变量 message，然后再定义一个函数，在该函数内输出全局变量 message 的值，代码如下：

```
01 message = '唯有在被追赶的时候，你才能真正地奔跑。'    # 全局变量
02 def f_demo():
03     print('函数体内：全局变量 message =',message)  # 在函数体内输出全局变量的值
04 f_demo()                                          # 调用函数
05 print('函数体外：全局变量 message =',message)      # 在函数体外输出全局变量的值
```

运行上面的代码，将显示以下内容：

```
函数体内：全局变量 message = 唯有在被追赶的时候，你才能真正地奔跑。
函数体外：全局变量 message = 唯有在被追赶的时候，你才能真正地奔跑。
```

> **说明：**当局部变量与全局变量重名时，对函数体内的变量进行赋值后，不影响函数体外的变量。

（2）如果一个变量在函数体内定义，并且使用 global 关键字修饰后，该变量也就变为全局

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

变量。在函数体外可以访问该变量，并且在函数体内还可以对其进行修改。

例如，定义两个同名的全局变量和局部变量，并输出它们的值，代码如下：

```
message = '唯有在被追赶的时候，你才能真正地奔跑。'          # 全局变量
print('函数体外：message =',message)                    # 在函数体外输出全局变量的值
def f_demo():
    message = '命运给予我们的不是失望之酒，而是机会之杯。'# 局部变量
    print('函数体内：message =',message)                 # 在函数体内输出局部变量的值
f_demo()  # 调用函数
print('函数体外：message =',message)                    # 在函数体外输出全局变量的值
```

执行上面的代码后，将显示以下内容：

```
函数体外：message = 唯有在被追赶的时候，你才能真正地奔跑。
函数体内：message = 命运给予我们的不是失望之酒，而是机会之杯。
函数体外：message = 唯有在被追赶的时候，你才能真正地奔跑。
```

从上面的结果中可以看出，在函数内部定义的变量即使与全局变量重名，也不影响全局变量的值。那么想要在函数体内部改变全局变量的值，需要在定义局部变量时，使用 global 关键字修饰。例如，将上面的代码修改为以下内容：

```
message = '唯有在被追赶的时候，你才能真正地奔跑。'          # 全局变量
print('函数体外：message =',message)                    # 在函数体外输出全局变量的值
def f_demo():
    global message                                      # 将 message 声明为全局变量
    message = '命运给予我们的不是失望之酒，而是机会之杯。'# 全局变量
    print('函数体内：message =',message)                 # 在函数体内输出全局变量的值
f_demo()  # 调用函数
print('函数体外：message =',message)                    # 在函数体外输出全局变量的值
```

执行上面的代码后，将显示以下内容：

```
函数体外：message = 唯有在被追赶的时候，你才能真正地奔跑。
函数体内：message = 命运给予我们的不是失望之酒，而是机会之杯。
函数体外：message = 命运给予我们的不是失望之酒，而是机会之杯。
```

从上面的结果中可以看出，在函数体内部修改了全局变量的值。

> **注意：**尽管 Python 允许全局变量和局部变量重名，但是在实际开发时，不建议这么做，因为这样容易让代码混乱，很难分清哪些是全局变量，哪些是局部变量。

8.5 匿名函数（lambda）

匿名函数是指没有名字的函数，应用在需要一个函数，但是又不想费神去命名这个函数的场合。在通常情况下，这样的函数只使用一次。在 Python 中，使用 lambda 表达式创建匿名函数，其语法格式如下：

```
result = lambda [arg1 [,arg2,…,argn]]:expression
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

参数说明如下。

- result：用于调用 lambda 表达式。
- [arg1 [,arg2,⋯,argn]]：可选参数，用于指定要传递的参数列表，多个参数间使用逗号“,”进行分隔。
- expression：必选参数，用于指定一个实现具体功能的表达式。如果有参数，那么在该表达式中将应用这些参数。

> **注意**：使用 lambda 表达式时，参数可以有多个，用逗号“,”进行分隔，但是表达式只能有一个，即只能返回一个值，而且也不能出现其他非表达式语句（如 for 或 while）。

例如，要定义一个计算圆面积的函数，常规的代码如下：

```
01 import math                      # 导入 math 模块
02 def circlearea(r):               # 计算圆的面积的函数
03     result = math.pi*r*r         # 计算圆的面积
04     return result                # 返回圆的面积
05 r = 10                           # 半径
06 print('半径为',r,'的圆面积为: ',circlearea(r))
```

执行上面的代码后，将显示以下内容：

```
半径为 10 的圆面积为:  314.1592653589793
```

使用 lambda 表达式的代码如下：

```
01 import math                      # 导入 math 模块
02 r = 10                           # 半径
03 result = lambda r:math.pi*r*r    # 计算圆的面积的 lambda 表达式
04 print('半径为',r,'的圆面积为: ',result(r))
```

执行上面的代码后，将显示以下内容：

```
半径为 10 的圆面积为:  314.1592653589793
```

从上面的示例中可以看出，虽然使用 lambda 表达式比使用自定义函数的代码减少了一些，但是在使用 lambda 表达式时，需要定义一个变量，用于调用该 lambda 表达式，否则将输出类似的结果。

```
<function <lambda> at 0x0000000002FDD510>
```

这看似有点画蛇添足。那么 lambda 表达式具体应该怎么应用呢？实际上，lambda 的首要用途是指定短小的回调函数。

8.6　快学快用

（1）时光荏苒，日月如梭，不知不觉已经来到 2019 年。请编写一个小程序，实现输入任意年月日，计算出这是这一年的第几天。如输入 20190203，则提示这是 2019 年的第 34 天。

程序参考代码如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
 def whichday(m,d):
if m in (1,3,5,7):
return u'这是今年第'+str((m-1)*30+(m-1)/2+d-2)+u'天'
if m in (2,4,6,8,10,12):
return u'这是今年第'str((m-1)*30+m/2+d-2)+u'天'
if m in (9,11):
return u'这是今年第'str((m-1)*30+(m+1)/2+d-2)+u'天
```

（2）下面 list 中提供了几组 IP 地址，编写一个函数，将 list 中重复出现的 IP 地址找出来。

```
list=['192.168.1.1','10.192.66.72','10.199.88.132','192.168.1.1','1.192.168.163']
```

（3）编写一个函数，把输入的字母统一成大写模式，如输入“go big or go home”，输出为“GO BIG OR GO HOME”。

（4）Python 提供了很多有用的内置函数或模块，大家熟悉一下这些函数或模块的用途，在实际开发时如果遇到相关应用，可以联想到相关函数或模块，然后详细学习并使用，事半功倍。集合类操作内置函数如表 8.1 所示。如果要详细了解这些函数或模块的使用，可以扫描二维码，到快用云端支持平台学习、训练、实战。

表 8.1　集合类操作内置函数（2）

dict([arg])	创建数据字典
list([iterable])	将一个集合类转换为另外一个集合类
set()	set 对象实例化
frozenset([iterable])	产生一个不可变的 set
str([object])	转换为 string 类型
sorted(iterable[, cmp[, key[, reverse]]])	队集合排序
tuple([iterable])	生成一个 tuple 类型
xrange([start], stop[, step])	xrange()函数与 range()函数类似，但 xrange()函数并不创建列表，而是返回一个 xrange 对象，它的行为与列表相似，但是只在需要时才计算列表值，当列表很大时，这个特性能为我们节省内存

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 9 章

面向对象程序设计

面向对象程序设计是在面向过程程序设计的基础上发展而来的，它比面向过程编程具有更强的灵活性和扩展性。面向对象程序设计也是一个程序员发展的“分水岭”，很多初学者和略有成就的开发者，就是因为无法理解“面向对象”而放弃。这里想提醒一下初学者：要想在编程这条路上走得比别人远，就一定要掌握面向对象编程技术。

Python 从设计之初就已经是一门面向对象的语言了，它可以很方便地创建类和对象。本章将对面向对象程序设计进行详细讲解。

9.1 面向对象概述

面向对象（Object Oriented）的英文缩写是 OO，它是一种设计思想。从 20 世纪 60 年代提出面向对象的概念到现在，它已经发展成为一种比较成熟的编程思想，并且逐步成为目前软件开发领域的主流技术。如我们经常听说的面向对象编程（Object Oriented Programming，OOP）就是主要针对大型软件设计而提出的，它可以使软件设计更加灵活，并且能更好地进行代码复用。

面向对象中的对象（Object）通常是指客观世界中存在的对象，这个对象具有唯一性，对象之间各不相同，各有各的特点，每一个对象都有自己的运动规律和内部状态；对象与对象之间又是可以相互联系、相互作用的。另外，对象也可以是一个抽象的事物，例如，可以从圆形、正方形、三角形等图形中抽象出一个简单图形，简单图形就是一个对象，它有自己的属性和行为，图形中边的个数是它的属性，图形的面积也是它的属性，输出图形的面积就是它的行为。概括地讲，面向对象技术是一种从组织结构上模拟客观世界的方法。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

9.1.1 对象

对象是一个抽象概念，英文称作“Object”，表示任意存在的事物。世间万物皆对象！在现实世界中，随处可见的一种事物就是对象，对象是事物存在的实体，如一个人。

通常将对象划分为两个部分，即静态部分与动态部分。静态部分被称为“属性”，任何对象都具备自身属性，这些属性不仅是客观存在的，而且是不能被忽视的，如人的性别；动态部分指的是对象的行为，即对象执行的动作，如人可以行走。

> **说明：**在 Python 中，一切都是对象，即不仅把具体的事物称为对象，字符串、函数等也都是对象。这说明 Python 天生就是面向对象的。

9.1.2 类

类是封装对象的属性和行为的载体，反过来说具有相同属性和行为的一类实体被称为类。例如，把雁群比作大雁类，那么大雁类就具备了喙、翅膀和爪等属性，觅食、飞行和睡觉等行为，而一只要从北方飞往南方的大雁则被视为大雁类的一个对象。

在 Python 语言中，类是一种抽象概念，如定义一个大雁类（Geese），在该类中，可以定义每个对象共有的属性和方法；而一只要从北方飞往南方的大雁则是大雁类的一个对象（wildGeese），对象是类的实例。有关类的具体实现将在 9.2 节进行详细介绍。

9.1.3 面向对象程序设计的特点

面向对象程序设计具有三大基本特征：封装、继承和多态，下面分别进行描述。

1．封装

封装是面向对象编程的核心思想，将对象的属性和行为封装起来，其载体就是类，类通常会对客户隐藏其实现细节，这就是封装的思想。例如，用户使用计算机，只需要使用手指敲击键盘就可以实现一些功能，而无须知道计算机内部是如何工作的。

采用封装思想保证了类内部数据结构的完整性，使用该类的用户不能直接看到类中的数据结构，而只能执行类允许公开的数据，这样就避免了外部对内部数据的影响，提高了程序的可维护性。

2．继承

矩形、菱形、平行四边形和梯形等都是四边形。因为四边形与它们具有共同的特征：拥有 4 个边。只要将四边形适当延伸，就会得到上述图形。以平行四边形为例，如果把平行四边形看作四边形的延伸，那么平行四边形就复用了四边形的属性和行为，同时添加了平行四边形特有的属性和行为，如平行四边形的对边平行且相等。在 Python 中，可以把平行四边形类看作是继承四边形类后产生的类，其中，将类似于平行四边形的类称为子类，将类似于四边形的类称为父类或超类。值得注意的是，在阐述平行四边形和四边形的关系时，可以说平行四边形是

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

特殊的四边形，但不能说四边形是平行四边形。同理，Python 中可以说子类的实例都是父类的实例，但不能说父类的实例是子类的实例。

综上所述，继承是实现重复利用的重要手段，子类通过继承复用了父类的属性和行为的同时，又添加了子类特有的属性和行为。

3．多态

将父类对象应用于子类的特征就是多态。例如，创建一个螺丝类，螺丝类有两个属性：粗细和螺纹密度；然后再创建两个类，一个是长螺丝类，另一个是短螺丝类，并且它们都继承了螺丝类。这样长螺丝类和短螺丝类不仅具有相同的特征（粗细相同，且螺纹密度也相同），还具有不同的特征（一个长，一个短，长的可以用来固定大型支架，短的可以固定生活中的家具）。综上所述，一个螺丝类衍生出不同的子类，子类继承父类特征的同时，也具备了自己的特征，并且能够实现不同的效果，这就是多态化的结构。

9.2　类的定义和使用

在 Python 中，类表示具有相同属性和方法的对象的集合。在使用类时，需要先定义类，然后再创建类的实例，通过类的实例就可以访问类中的属性和方法了。下面进行具体介绍。

9.2.1　定义类

在 Python 中，类的定义使用 class 关键字来实现，语法如下：

```
class ClassName:
    '''类的帮助信息'''        # 类文档字符串
    statement               # 类体
```

参数说明如下。

- ClassName：用于指定类名，一般使用大写字母开头，如果类名中包括两个单词，第二个单词的首字母也大写，这种命名方法也称为“驼峰式命名法”，这是惯例。当然，也可根据自己的习惯命名，但是一般推荐按照惯例来命名。
- '''类的帮助信息'''：用于指定类的文档字符串，定义该字符串后，在创建类的对象时，输入类名和左侧的括号“(”后，将显示该信息。
- statement：类体，主要由类变量（或类成员）、方法和属性等定义语句组成。如果在定义类时，没想好类的具体功能，也可以在类体中直接使用 pass 语句代替。

例如，下面以大雁为例声明一个类，代码如下：

```
01 class Geese:
02     '''大雁类'''
03     pass
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

9.2.2 创建类的实例

定义完类后，并不会真正创建一个实例，这有点像一个汽车的设计图，设计图可以告诉你汽车看上去怎么样，但设计图本身不是一个汽车。你不能开走它，它只能用来建造真正的汽车，而且可以使用它制造很多汽车。那么如何创建实例呢？

class 语句本身并不创建类的任何实例，所以在类定义完成以后，可以创建类的实例，即实例化类的对象。创建类的实例的语法如下：

```
ClassName(parameterlist)
```

其中，ClassName 是必选参数，用于指定具体的类；parameterlist 是可选参数，当创建一个类时，没有创建__init__()方法（该方法将在 9.2.3 节进行详细介绍），或者__init__()方法只有一个 self 参数时，parameterlist 可以省略。

例如，创建 9.2.1 节定义的 Geese 类的实例，可以使用下面的代码：

```
01 wildGoose = Geese()     # 创建大雁类的实例
02 print(wildGoose)
```

执行上面代码后，将显示类似下面的内容：

```
<__main__.Geese object at 0x0000000002F47AC8>
```

从上面的执行结果中可以看出，wildGoose 是 Geese 类的实例。

> **注意：**在 Python 中创建实例不使用 new 关键字，这是它与其他面向对象语言的区别。

9.2.3 创建__init__()方法

在创建类后，类通常会自动创建一个__init__()方法，该方法是一个特殊的方法，类似 Java 语言中的构造方法。每当创建一个类的新实例时，Python 都会自动执行它。__init__()方法必须包含一个 self 参数，并且必须是第一个参数。self 参数是一个指向实例本身的引用，用于访问类中的属性和方法。在方法调用时会自动传递实际参数 self。因此，当__init__()方法只有一个参数时，在创建类的实例时，就不需要指定实际参数了。

> **说明：**在__init__()方法的名称中，开头和结尾处是两个下画线（中间没有空格），这是一种约定，旨在区分 Python 默认方法和普通方法。

例如，下面仍然以大雁为例声明一个类，并且创建__init__()方法，代码如下：

```
01 class Geese:
02     '''大雁类'''
03     def __init__(self):          # 构造方法
04         print("我是大雁类! ")
05 wildGoose = Geese()              # 创建大雁类的实例
```

运行上面的代码，将输出以下内容：

```
我是大雁类!
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

从上面的运行结果中可以看出，在创建大雁类的实例时，虽然没有为__init__()方法指定参数，但是该方法会自动执行。

常见错误

在为类创建__init__()方法时，在开发环境中运行下面的代码：

```
01 class Geese:
02     '''大雁类'''
03     def __init__():                # 构造方法
04         print("我是大雁类！")
05 wildGoose = Geese()                # 创建大雁类的实例
```

将显示如图 9.1 所示的异常信息，该错误的解决方法是在第 3 行代码的括号中添加 self。

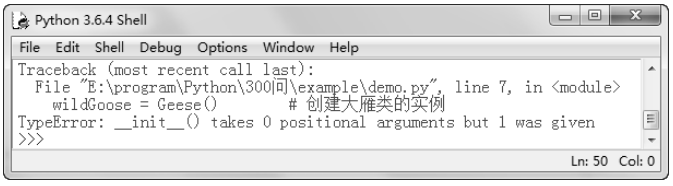


图 9.1　缺少 self 参数抛出的异常信息

在__init__()方法中，除了 self 参数外，还可以自定义一些参数，参数间使用逗号“,”进行分隔。例如，下面的代码将在创建__init__()方法时，再指定 3 个参数，分别是 beak、wing 和 claw。

```
01 class Geese:
02     '''大雁类'''
03     def __init__(self,beak,wing,claw):          # 构造方法
04         print("我是大雁类！我有以下特征：")
05         print(beak)                              # 输出喙的特征
06         print(wing)                              # 输出翅膀的特征
07         print(claw)                              # 输出爪子的特征
08 beak_1 = "喙的基部较高，长度和头部的长度几乎相等"    # 喙的特征
09 wing_1 = "翅膀长而尖"                             # 翅膀的特征
10 claw_1 = "爪子是蹼状的"                           # 爪子的特征
11 wildGoose = Geese(beak_1,wing_1,claw_1)          # 创建大雁类的实例
```

执行上面的代码，将显示如图 9.2 所示的运行结果。

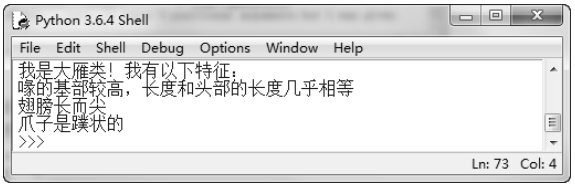


图 9.2　创建__init__()方法时，指定 3 个参数

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

9.2.4 创建类的成员并访问

类的成员主要由实例方法和数据成员组成。在类中创建了类的成员后，可以通过类的实例进行访问。下面进行详细介绍。

1．创建实例方法并访问

所谓实例方法是指在类中定义的函数，该函数是一种在类的实例上操作的函数。同__init__()方法一样，实例方法的第一个参数必须是 self，并且必须包含一个 self 参数。创建实例方法的语法格式如下：

```
def functionName(self,parameterlist):
    block
```

参数说明如下。

- functionName：用于指定方法名，一般使用小写字母开头。
- self：必要参数，表示类的实例，其名称可以是 self 以外的单词，使用 self 只是一个惯例而已。
- parameterlist：用于指定除 self 参数以外的参数，各参数间使用逗号“,”进行分隔。
- block：方法体，实现的具体功能。

> **说明：**实例方法和 Python 中的函数的主要区别就是，函数实现的是某个独立的功能，而实例方法是实现类中的一个行为，是类的一部分。

实例方法创建完成后，可以通过类的实例名称和点“.”操作符进行访问。具体的语法格式如下：

```
instanceName.functionName(parametervalue)
```

其中，instanceName 为类的实例名称；functionName 为要调用的方法名称；parametervalue 表示为方法指定对应的实际参数，其值的个数为 parameterlist 的个数减 1。

2．创建数据成员并访问

数据成员是指在类中定义的变量，即属性，根据定义位置，又可以分为类属性和实例属性。下面分别进行介绍。

（1）类属性

类属性是指定义在类中，并且在函数体外的属性。类属性可以在类的所有实例之间共享值，也就是在所有实例化的对象中公用。

> **说明：**类属性可以通过类名或者实例名访问。

例如，定义一个雁类 Geese，在该类中定义 3 个类属性，用于记录雁类的特征，代码如下：

```
01 class Geese:
02     '''雁类'''
03     neck = "脖子较长"                    # 定义类属性（脖子）
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    wing = "振翅频率高"                    # 定义类属性（翅膀）
    leg = "腿位于身体的中心支点，行走自如"   # 定义类属性（腿）
    def __init__(self):                    # 实例方法（相当于构造方法）
        print("我属于雁类！我有以下特征：")
        print(Geese.neck)                  # 输出脖子的特征
        print(Geese.wing)                  # 输出翅膀的特征
        print(Geese.leg)                   # 输出腿的特征
```

创建上面的类 Geese，然后创建该类的实例，代码如下：

```
geese = Geese()                            # 实例化一个雁类的对象
```

应用上面的代码创建 Geese 类的实例后，将显示以下内容：

```
我属于雁类！我有以下特征：
脖子较长
振翅频率高
腿位于身体的中心支点，行走自如
```

（2）实例属性

实例属性是指定义在类的方法中的属性，只作用于当前实例中。

例如，定义一个雁类 Geese，在该类的__init__()方法中定义 3 个实例属性，用于记录雁类的特征，代码如下：

```
class Geese:
    '''雁类'''
    def __init__(self):                              # 实例方法（相当于构造方法）
        self.neck = "脖子较长"                        # 定义实例属性（脖子）
        self.wing = "振翅频率高"                      # 定义实例属性（翅膀）
        self.leg = "腿位于身体的中心支点，行走自如"    # 定义实例属性（腿）
        print("我属于雁类！我有以下特征：")
        print(self.neck)                             # 输出脖子的特征
        print(self.wing)                             # 输出翅膀的特征
        print(self.leg)                              # 输出腿的特征
```

创建上面的类 Geese，然后创建该类的实例，代码如下：

```
geese = Geese()                                      # 实例化一个雁类的对象
```

应用上面的代码创建 Geese 类的实例后，将显示以下内容。

```
我属于雁类！我有以下特征：
脖子较长
振翅频率高
腿位于身体的中心支点，行走自如
```

> **说明：**实例属性只能通过实例名访问。如果通过类名访问实例属性，将抛出如图 9.3 所示的异常。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

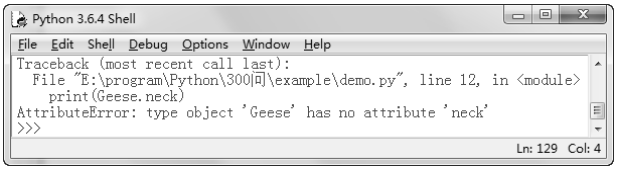


图 9.3　通过类名访问实例属性抛出异常

实例属性也可以通过实例名称修改，与类属性不同，通过实例名称修改实例属性后，并不影响该类的另一个实例中相应的实例属性的值。例如，定义一个雁类，并在__init__()方法中定义一个实例属性，然后创建两个 Geese 类的实例，并且修改第一个实例的属性，最后分别输出两个实例属性，代码如下：

```
class Geese:
    '''雁类'''
    def __init__(self):                          # 实例方法（相当于构造方法）
        self.neck = "脖子较长"                    # 定义实例属性（脖子）
        print(self.neck)                         # 输出脖子的特征
goose1 = Geese()                                 # 创建 Geese 类的实例 1
goose2 = Geese()                                 # 创建 Geese 类的实例 2
goose1.neck = "脖子没有天鹅的长"                  # 修改实例属性
print("goose1 的 neck 属性：",goose1.neck)
print("goose2 的 neck 属性：",goose2.neck)
```

运行上面的代码，将显示以下内容：

```
脖子较长
脖子较长
goose1 的 neck 属性： 脖子没有天鹅的长
goose2 的 neck 属性： 脖子较长
```

9.2.5　访问限制

在类的内部可以定义属性和方法，而在类的外部则可以直接调用属性或方法来操作数据，从而隐藏了类内部的复杂逻辑。但是 Python 并没有对属性和方法的访问权限进行限制。为了保证类内部的某些属性或方法不被外部所访问，可以在属性或方法名前面添加单下画线（_foo）、双下画线（__foo）或首尾加双下画线（__foo__），从而限制访问权限。其中，首尾双下画线、单下画线和双下画线的作用如下。

- __foo__：首尾双下画线表示定义特殊方法，一般是系统定义名字，如__init__()。
- _foo：以单下画线开头的表示 protected（保护）类型的成员，只允许类本身和子类进行访问，但不能使用“from module import *”语句导入。

例如，创建一个 Swan 类，定义保护属性_neck_swan，并在__init__()方法中访问该属性，然后创建 Swan 类的实例，并通过实例名输出保护属性_neck_swan，代码如下：

```
class Swan:
    '''天鹅类'''
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
03     _neck_swan = '天鹅的脖子很长'                    # 定义私有属性
04     def __init__(self):
05         print("__init__():", Swan._neck_swan)    # 在实例方法中访问私有属性
06 swan = Swan()                                    # 创建 Swan 类的实例
07 print("直接访问:" , swan._neck_swan)               # 保护属性可以通过实例名访问
```

执行上面的代码，将显示以下内容：

```
__init__(): 天鹅的脖子很长
直接访问: 天鹅的脖子很长
```

从上面的运行结果中可以看出，保护属性可以通过实例名访问。

- __foo：双下画线表示 private（私有）类型的成员，只允许定义该方法的类本身进行访问，而且也不能通过类的实例进行访问，但是可以通过“类的实例名.类名__xxx”方式访问。

例如，创建一个 Swan 类，定义私有属性__neck_swan，并在__init__()方法中访问该属性，然后创建 Swan 类的实例，并通过实例名输出私有属性__neck_swan，代码如下：

```
01 class Swan:
02     '''天鹅类'''
03     __neck_swan = '天鹅的脖子很长'                    # 定义私有属性
04     def __init__(self):
05         print("__init__():", Swan.__neck_swan)   # 在实例方法中访问私有属性
06 swan = Swan()                                    # 创建 Swan 类的实例
07 # 私有属性，可以通过“实例名.类名__xxx”方式访问
08 print("加入类名:" , swan._Swan__neck_swan)
09 print("直接访问:" , swan.__neck_swan)              # 私有属性不能通过实例名访问,出错
```

执行上面的代码后，将输出如图 9.4 所示的结果。

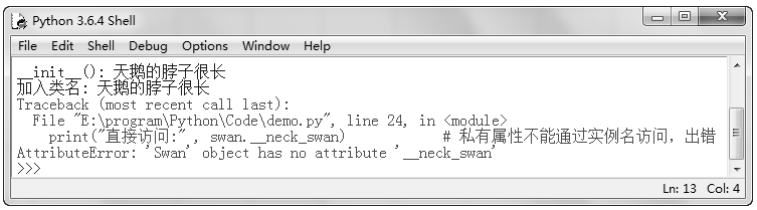


图 9.4　访问私有属性

从上面的运行结果中可以看出：私有属性可以通过类名.属性名访问，也可以通过“实例名.类名__xxx”方式访问，但是不能直接通过实例名.属性名访问。

9.3　属性（property）

本节介绍的属性与 9.2.4 节介绍的类属性和实例属性不同。9.2.4 节介绍的属性将返回所存储的值，而本节要介绍的属性则是一种特殊的属性，访问它时将计算它的值。另外，该属性还可以为属性添加安全保护机制。下面分别进行介绍。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

9.3.1 创建用于计算的属性

在 Python 中，可以通过@property（装饰器）将一个方法转换为属性，从而实现用于计算的属性。将方法转换为属性后，可以直接通过方法名来访问方法，而不需要再添加一对小括号“()”，这样可以让代码更加简洁。

通过@property 创建用于计算的属性的语法格式如下：

```
@property
def methodname(self):
    block
```

参数说明如下。

- methodname：用于指定方法名，一般使用小写字母开头。该名称最后将作为创建的属性名。
- self：必要参数，表示类的实例。
- block：方法体，实现的具体功能。在方法体中，通常以 return 语句结束，用于返回计算结果。

例如，定义一个矩形类，在__init__()方法中定义两个实例属性，然后再定义一个计算矩形面积的方法，并应用@property 将其转换为属性，最后创建类的实例，并访问转换后的属性，代码如下：

```
01 class Rect:
02     def __init__(self,width,height):
03         self.width = width                # 矩形的宽
04         self.height = height              # 矩形的高
05     @property                             # 将方法转换为属性
06     def area(self):                       # 计算矩形的面积的方法
07         return self.width*self.height     # 返回矩形的面积
08 rect = Rect(800,600)                      # 创建类的实例
09 print("面积为：",rect.area)                # 输出属性的值
```

运行上面的代码，将显示以下运行结果：

```
面积为： 480000
```

> **注意：**通过@property 转换后的属性不能重新赋值，如果对其重新赋值，将抛出如图 9.5 所示的异常信息。

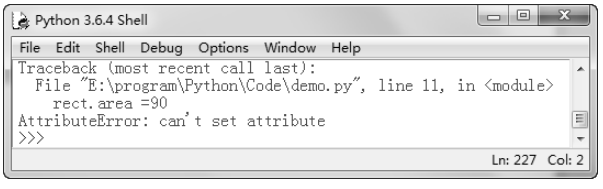


图 9.5 AttributeError 异常

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

9.3.2　为属性添加安全保护机制

在 Python 中，在默认情况下，创建的类属性或者实例是可以在类体外进行修改的，如果想要限制其不能在类体外修改，可以将其设置为私有的，但设置为私有后，在类体外也不能获取它的值。如果想要创建一个可以读取，但不能修改的属性，那么可以使用@property 实现只读属性。

例如，创建一个电视节目类 TVshow，再创建一个 show 属性，用于显示当前播放的电视节目，代码如下：

```
01 class TVshow:   # 定义电视节目类
02     def __init__(self,show):
03         self.__show = show
04     @property                          # 将方法转换为属性
05     def show(self):                    # 定义 show()方法
06         return self.__show             # 返回私有属性的值
07 tvshow = TVshow("正在播放《战狼 2》") # 创建类的实例
08 print("默认：",tvshow.show)            # 获取属性值
```

执行上面的代码，将显示以下内容：

```
默认：  正在播放《战狼 2》
```

通过上面的方法创建的 show 属性是只读的，尝试修改该属性的值，再重新获取。在上面代码的下方添加以下代码：

```
01 tvshow.show = "正在播放《红海行动》"   # 修改属性值
02 print("修改后：",tvshow.show)          # 获取属性值
```

运行后，将显示如图 9.6 所示的运行结果。其中出现的异常信息就是修改属性 show 时抛出的异常。

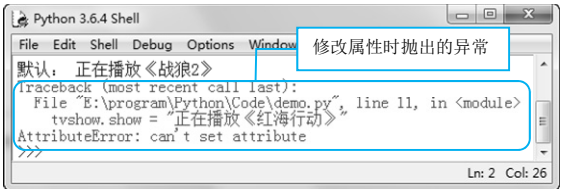


图 9.6　修改只读属性时抛出的异常

不仅可以将属性设置为只读属性，而且可以为属性设置拦截器，即允许对属性进行修改，但修改时需要遵守一定的约束。

9.4　继承

在编写类时，并不是每次都要从空白开始。当要编写的类和另一个已经存在的类之间存在一定的继承关系时，就可以通过继承来达到代码重用的目的，提高开发效率。下面将介绍如何在 Python 中实现继承。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

9.4.1 继承的基本语法

继承是面向对象编程最重要的特性之一，它源于人们认识客观世界的过程，是自然界普遍存在的一种现象。例如，我们每一个人都从祖辈和父母那里继承了一些体貌特征，但是每个人却又不同于父母，因为每个人都存在自己的一些特性，这些特性是独有的，在父母身上并没有体现。在程序设计中实现继承，表示这个类拥有它继承的类的所有公有成员或者受保护成员。在面向对象编程中，被继承的类称为父类或基类，新的类称为子类或派生类。

通过继承不仅可以实现代码的重用，还可以通过继承来理顺类与类之间的关系。在 Python 中，可以在类定义语句中的类名右侧使用一对小括号将要继承的基类名称括起来，从而实现类的继承。具体的语法格式如下：

```
class ClassName(baseclasslist):
    '''类的帮助信息'''          # 类文档字符串
    statement                  # 类体
```

参数说明如下。

- ClassName：用于指定类名。
- baseclasslist：用于指定要继承的基类，可以有多个，类名之间用逗号“,”进行分隔。如果不指定，将使用所有 Python 对象的根类 object。
- '"类的帮助信息"'：用于指定类的文档字符串，定义该字符串后，在创建类的对象时，输入类名和左侧的括号“(”后，将显示该信息。
- statement：类体，主要由类变量（或类成员）、方法和属性等定义语句组成。如果在定义类时，没想好类的具体功能，也可以在类体中直接使用 pass 语句代替。

9.4.2 方法重写

基类的成员都会被派生类继承，当基类中的某个方法不完全适用于派生类时，就需要在派生类中重写父类的这个方法，这和 Java 语言中的方法重写是一样的。

在基类中定义一个 harvest()方法，无论派生类是什么水果，都显示“水果……”，如果想要针对不同水果给出不同的提示，可以在派生类中重写 harvest()方法。例如，在创建派生类 Orange 时，重写 harvest()方法的代码如下：

```
01 class Orange(Fruit):                              # 定义橘子类（派生类）
02     color = "橙色"
03     def __init__(self):
04         print("\n我是橘子")
05     def harvest(self, color):
06         print("橘子是: " + color + "的! ")     # 输出的是形式参数 color
07         print("橘子已经收获……")
08         print("橘子原来是: " + Fruit.color + "的! ");  # 输出的是类属性 color
```

添加 harvest()方法后，再次运行代码，将显示如图 9.7 所示的运行结果。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 9.7　重写 Orange 类的 harvest()方法的结果

9.4.3　派生类中调用基类的__init__()方法

在派生类中定义__init__()方法时，不会自动调用基类的__init__()方法。例如，定义一个 Fruit 类，在__init__()方法中创建类属性 color，然后在 Fruit 类中定义一个 harvest()方法，在该方法中输出类属性 color 的值，再创建继承自 Fruit 类的 Apple 类，最后创建 Apple 类的实例，并调用 harvest()方法，代码如下：

```
01 class Fruit:                      # 定义水果类（基类）
02     def __init__(self,color = "绿色"):
03         Fruit.color = color       # 定义类属性
04     def harvest(self):
05         print("水果原来是：" + Fruit.color + "的！");  # 输出的是类属性 color
06 class Apple(Fruit):               # 定义苹果类（派生类）
07     def __init__(self):
08         print("我是苹果")
09 apple = Apple()                   # 创建类的实例（苹果）
10 apple.harvest()                   # 调用基类的 harvest()方法
```

执行上面的代码后，将显示如图 9.8 所示的异常信息。

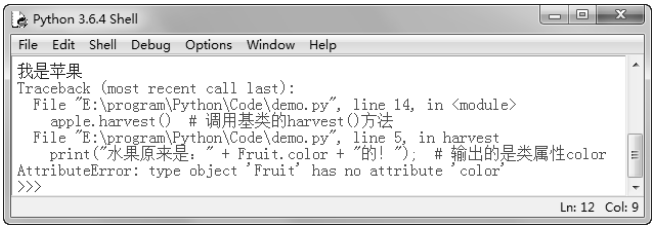


图 9.8　基类的__init__()方法未执行引起的异常

因此，要让派生类调用基类的__init__()方法进行必要的初始化，需要在派生类中使用 super()函数调用基类的__init__()方法。例如，在上面的第 8 行代码的下方添加以下代码：

```
super().__init__()                   # 调用基类的__init__()方法
```

注意：在添加上面的代码时，一定要注意缩进的正确性。

运行后将显示以下正常的运行结果。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
我是苹果
水果原来是：绿色的!
```

9.5 快学快用

（1）F-22 战斗机是美国一型单座双发高隐身性第五代战斗机。根据 F-22 的下列信息，封装成一个 fighter 类，包括名称、编号、代号、厂家等信息，并在 display()方法中显示。

- name（名字）：F-22 fighter。
- Number（编号）：F-22。
- Code（代号）：Raptor（猛禽）。
- price（价格）：1.5 亿美元。
- vender（厂家）：Lockheed。

程序参考代码如下：

```
class fighter():
    name = None
    Number = None
    price = 0
    vender = None
    def display(self):
        print "name:", self.name
        print "Number:, self.name
        print "Code:", self.code
        print "price:", self.price
        print "vender:", self.vender

Jasin = fighte()
Jasin.name = "F-22 fighter"
Jasin.Number = "F-22"
Jasin.Code = "Raptor"
Jasin.price ="1.5 亿美元"
Jasin.vender ="Lockheed;"
Jasin.display()
```

输出结果如下：

```
Name: F-22 fighter
Number: F-22
Code: Raptor
price: 1.5 亿美元
vender: Lockheed
```

（2）定义宠物类型，创建多个宠物，如狗（dog）、兔子（rabbit）、鹦鹉（parrot）、猫（cat）、仓鼠（hamster）、乌龟（tortoise）等，展示每个宠物的信息，如颜色、体重等信息。

（3）以下代码的调试结果为（　　）。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
class Square {
    int a;
    void Square() {
        a = 10;
    }
    public static void main(String[] args) {
        Square s = new Square();
        System.out.println(s.a);
    }
}
```

A．输出 10　　B．编译错误　　C．输出 0　　D．运行错误

（4）Python 提供了很多有用的内置函数或模块，大家熟悉一下这些函数或模块的用途，在实际开发时如果遇到相关应用，可以联想到相关函数或模块，然后详细学习并使用，事半功倍。反射内置函数如表 9.1 所示。如果要详细了解这些函数或模块的使用，可以扫描二维码，到快用云端支持平台学习、训练、实战。

表 9.1　反射内置函数(1)

callable(object)	检查对象 object 是否可调用。 • 类是可以被调用的 • 实例是不可以被调用的，除非类中声明了__call__方法
classmethod()	• 注解，用来说明这个方式是一个类方法 • 类方法既可以被类调用，也可以被实例调用 • 类方法类似于 Java 中的 static 方法 • 类方法中不需要有 self 参数
compile(source, filename, mode[, flags[, dont_inherit]])	将 source 编译为代码或者 AST 对象。代码对象能够通过 exec 语句来执行或者通过 eval()进行求值。 • 参数 source：字符串或者 AST（Abstract Syntax Trees）对象 • 参数 filename：代码文件名称，如果不是从文件读取代码，则传递一些可辨认的值 • 参数 mode：指定编译代码的种类。可以指定为'exec'、'eval'、'single' • 参数 flags 和 dont_inherit：这两个参数暂不介绍
dir([object])	• 不带参数时，返回当前范围内的变量、方法和定义的类型列表 • 带参数时，返回参数的属性、方法列表 • 如果参数包含__dir__()方法，该方法将被调用 • 如果参数不包含__dir__()方法，该方法将最大限度地收集参数信息
delattr(object, name)	删除 object 对象名为 name 的属性

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 3 篇　高 级 篇

第 10 章

模　块

Python 提供了强大的模块支持，主要体现为不仅在 Python 标准库中包含了大量的模块（称为标准模块），而且还有很多第三方模块，另外，开发者自己也可以开发自定义模块。通过这些强大的模块支持，将极大地提高我们的开发效率。

本章首先对如何开发自定义模块进行详细介绍，然后介绍如何使用标准模块和第三方模块。

10.1　模块概述

模块的英文是 Modules，可以认为是一盒（箱）主题积木，通过它可以拼出某一主题的东西。这与第 8 章介绍的函数不同，一个函数相当于一块积木，而一个模块中可以包括很多函数，也就是很多积木，所以也可以说模块相当于一盒积木。

在 Python 中，一个扩展名为.py 的文件就称为一个模块。例如，在第 8 章的实例中创建的 function_bmi.py 文件就是一个模块，如图 10.1 所示。

图 10.1　一个.py 文件就是一个模块

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

在通常情况下，我们把能够实现某一特定功能的代码放置在一个文件中作为一个模块，从而方便其他程序和脚本导入并使用。另外，使用模块也可以避免函数名和变量名冲突。

经过前面的学习，我们知道对于 Python 代码可以写到一个文件中。但是随着程序不断变大，为了便于维护，需要将其分为多个文件，这样可以提高代码的可维护性。另外，使用模块还可以提高代码的可重用性，即编写好一个模块后，只要是实现该功能的程序，都可以导入这个模块实现。

10.2　自定义模块

在 Python 中，自定义模块有两个作用，一个作用是规范代码，让代码更易于阅读；另一个作用是方便其他程序使用已经编写好的代码，提高开发效率。要实现自定义模块主要分为两部分，一部分是创建模块，另一部分是导入模块。下面分别进行介绍。

10.2.1　创建模块

创建模块可以将模块中相关的代码（变量定义和函数定义等）编写在一个单独的文件中，并且将该文件命名为“模块名+.py”的形式。

> **注意：**在创建模块时，设置的模块名不能是 Python 自带的标准模块名称。如果设置了相同的模块名称，那么在导入该模块后，Python 自带的标准模块就不能再导入了。

实例 01　**根据身高、体重计算 BMI 指数**

下面创建一个用于根据身高、体重计算 BMI 指数的模块，命名为 bmi.py，其中 bmi 为模块名，.py 为扩展名。关键代码如下：

```
def fun_bmi(person,height,weight):
    '''功能：根据身高和体重计算 BMI 指数
       person：姓名
       height：身高，单位：米
       weight：体重，单位：千克
    '''
    print(person + "的身高：" + str(height) + "米 \t 体重：" + str(weight) + "千克")
    bmi=weight/(height*height)      # 用于计算 BMI 指数，公式为“体重/身高的平方”
    print(person + "的 BMI 指数为："+str(bmi))   # 输出 BMI 指数
    # 此处省略了显示判断结果的代码
def fun_bmi_upgrade(*person):
    '''功能：根据身高和体重计算 BMI 指数（升级版）
       *person：可变参数，该参数中需要传递带 3 个元素的列表，
       分别为姓名、身高（单位：米）和体重（单位：千克）
    '''
    # 此处省略了函数主体代码
```

> **注意：**模块文件的扩展名必须是.py。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

10.2.2 使用import语句导入模块

创建模块后，就可以在其他程序中使用该模块了。要使用模块需要先以模块的形式加载模块中的代码，可以使用 import 语句实现。import 语句的基本语法格式如下：

```
import modulename [as alias]
```

其中，modulename 为要导入模块的名称；[as alias]为给模块起的别名，通过该别名也可以使用模块。

下面将导入实例 01 所编写的模块 bmi，并执行该模块中的函数。在模块文件 bmi.py 的同级目录下创建一个名称为 main.py 的文件，在该文件中导入模块 bmi，并且执行该模块中的 fun_bmi()函数，代码如下：

```
01 import bmi                          # 导入 bmi 模块
02 bmi.fun_bmi("尹一伊",1.75,120)  # 执行模块中的 fun_bmi()函数
```

执行上面的代码，将显示如图 10.2 所示的运行结果。

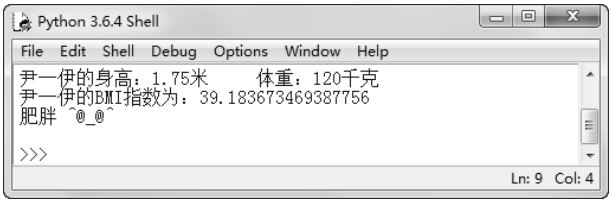


图 10.2　导入模块并执行模块中的函数

> **说明：**在调用模块中的变量、函数或者类时，需要在变量名、函数名或者类名前添加“模块名.”作为前缀。例如，上面代码中的 bmi.fun_bmi，则表示调用 bmi 模块中的 fun_bmi()函数。

多学两招

如果模块名比较长不容易记住，可以在导入模块时，使用 as 关键字为其设置一个别名，然后就可以通过这个别名来调用模块中的变量、函数和类等。例如，将上面导入模块的代码修改为以下内容：

```
import bmi as m                        # 导入 bmi 模块并设置别名为 m
```

然后，在调用 bmi 模块中的 fun_bmi()函数时，可以使用下面的代码：

```
m.fun_bmi("尹一伊",1.75,120)           # 执行模块中的 fun_bmi()函数
```

使用 import 语句还可以一次导入多个模块，在导入多个模块时，模块名之间使用逗号“,”进行分隔。例如，分别创建了 bmi.py、tips.py 和 differenttree.py 3 个模块文件。想要将这 3 个模块全部导入，可以使用下面的代码：

```
import bmi,tips,differenttree
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

10.2.3　使用from…import语句导入模块

在使用 import 语句导入模块时，每执行一条 import 语句都会创建一个新的命名空间（namespace），并且在该命名空间中执行与.py 文件相关的所有语句。在执行时，需在具体的变量、函数和类名前加上“模块名.”前缀。如果不想在每次导入模块时都创建一个新的命名空间，而是将具体的定义导入到当前的命名空间中，这时可以使用 from…import 语句。使用 from…import 语句导入模块后，不需要再添加前缀，直接通过具体的变量、函数和类名等访问即可。

> **说明：**命名空间可以理解为记录对象名字和对象之间对应关系的空间。目前 Python 的命名空间大部分都是通过字典（dict）来实现的。其中，key 是标识符；value 是具体的对象。例如，key 是变量的名字，value 则是变量的值。

from…import 语句的语法格式如下：

```
from modelname import member
```

参数说明如下。

- modelname：模块名称，区分字母大小写，需要和定义模块时设置的模块名称的大小写保持一致。
- member：用于指定要导入的变量、函数或者类等。可以同时导入多个定义，各个定义之间使用逗号“,”进行分隔。如果想导入全部定义，也可以使用通配符星号“*”代替。

多学两招

在导入模块时，如果使用通配符“*”导入全部定义后，想查看具体导入了哪些定义，可以通过显示 dir()函数的值来查看。例如，执行 print(dir())语句后将显示类似下面的内容：

```
['__annotations__', '__builtins__', '__doc__', '__file__', '__loader__',
'__name__', '__package__', '__spec__', 'change', 'getHeight', 'getWidth']
```

其中 change、getHeight 和 getWidth 就是我们导入的定义。

例如，通过下面的 3 条语句都可以从模块导入指定的定义：

```
01 from bmi import fun_bmi                    # 导入 bmi 模块的 fun_bmi 函数
02 # 导入 bmi 模块的 fun_bmi 和 fun_bmi_upgrade 函数
03 from bmi import fun_bmi,fun_bmi_upgrade
04 from bmi import *                          # 导入 bmi 模块的全部定义（包括变量和函数）
```

> **注意：**在使用 from…import 语句导入模块中的定义时，需要保证所导入的内容在当前的命名空间中是唯一的，否则将出现冲突，后导入的同名变量、函数或者类会覆盖先导入的，这时就需要使用 import 语句进行导入。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

10.2.4 模块搜索目录

当使用 import 语句导入模块时，在默认情况下，会按照以下顺序进行查找。

（1）在当前目录（执行的 Python 脚本文件所在目录）下查找。

（2）到 PYTHONPATH（环境变量）下的每个目录中查找。

（3）到 Python 的默认安装目录下查找。

以上各个目录的具体位置保存在标准模块 sys 的 sys.path 变量中。可以通过以下代码输出具体的目录：

```
01 import sys                # 导入标准模块 sys
02 print(sys.path)           # 输出具体目录
```

例如，在 IDLE 窗口中执行上面的代码，将显示如图 10.3 所示的结果。

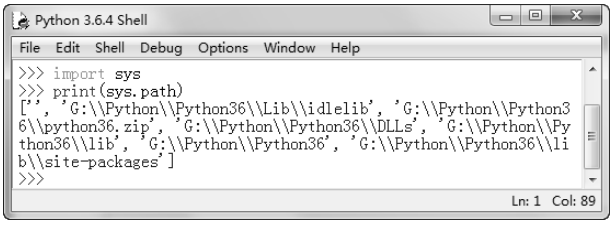


图 10.3　在 IDLE 窗口中查看具体目录

如果要导入的模块不在图 10.3 所示的目录中，那么在导入模块时，将显示如图 10.4 所示的异常。

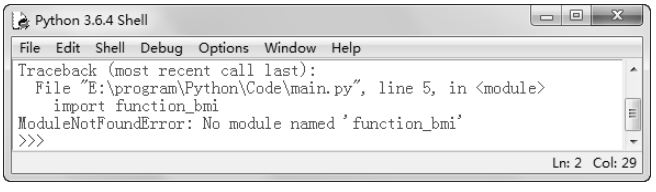

Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Traceback (most recent call last):
File "E:\program\Python\Code\main.py", line 5, in <module>
import function_bmi
ModuleNotFoundError: No module named 'function_bmi'
>>>
Ln: 2 Col: 29

图 10.4　找不到要导入的模块

注意：在使用 import 语句导入模块时，模块名是区分字母大小写的。

这时，我们可以通过以下 3 种方式添加指定的目录到 sys.path 中。

1．临时添加

临时添加即在导入模块的 Python 文件中添加。例如，需要将“E:\program\Python\Code\demo”目录添加到 sys.path 中，可以使用下面的代码：

```
01 import sys                                    # 导入标准模块 sys
02 sys.path.append('E:/program/Python/Code/demo')
```

执行上面的代码后，再输出 sys.path 的值，将得到以下结果：

```
['E:\\program\\Python\\Code', 'G:\\Python\\Python36\\python36.zip', 'G:\\Python\\Python36\\DLLs', 'G:\\Python\\Python36\\lib', 'G:\\Python\\Python36', 'G:\\Python\\Python36\\lib\\site-packages', 'E:/program/Python/Code/demo']
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

说明：通过该方法添加的目录只在执行当前文件的窗口中有效，窗口关闭后即失效。

2．增加.pth 文件（推荐）

在 Python 安装目录下的 Lib\site-packages 子目录中（例如，笔者的 Python 安装在 G:\Python\Python36 目录下，那么该路径为 G:\Python\Python36\Lib\site-packages），创建一个扩展名为.pth 的文件，文件名任意。这里创建一个 mrpath.pth 文件，在该文件中添加要导入模块所在的目录。例如，将模块目录“E:\program\Python\Code\demo”添加到 mrpath.pth 文件中，添加后的代码如下：

```
01 # .pth 文件是我创建的路径文件（这里为注释）
02 E:\program\Python\Code\demo
```

注意：创建.pth 文件后，需要重新打开要执行的导入模块的 Python 文件，否则新添加的目录不起作用。

说明：通过该方法添加的目录只在当前版本的 Python 中有效。

3．在 PYTHONPATH 环境变量中添加

打开“环境变量”对话框，如果没有 PYTHONPATH 系统环境变量，则需要先创建一个，否则直接选中 PYTHONPATH 变量，再单击“编辑”按钮，并且在弹出对话框的“变量值”文本框中添加新的模块目录，目录之前使用逗号进行分隔。例如，创建系统环境变量 PYTHONPATH，并指定模块所在目录为“E:\program\Python\Code\demo;”，效果如图 10.5 所示。

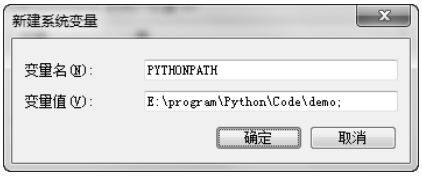


图 10.5　在环境变量中添加 PYTHONPATH 环境变量

注意：在环境变量中添加模块目录后，需要重新打开要执行的导入模块的 Python 文件，否则新添加的目录不起作用。

说明：通过该方法添加的目录可以在不同版本的 Python 中共享。

10.3　以主程序的形式执行

这里先来创建一个模块，名称为 christmastree。在该段代码中，首先定义一个全局变量，然后创建一个名称为 fun_christmastree()的函数，最后通过 print()函数输出一些内容，代码如下：

```
01 pinetree = '我是一棵松树'     # 定义一个全局变量（松树）
02 def fun_christmastree():    # 定义函数
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    '''功能：一个梦
       无返回值
    '''
    pinetree = '挂上彩灯、礼物……我变成一棵圣诞树 @^.^@ \n' # 定义局部变量
    print(pinetree)                                    # 输出局部变量的值
# ****************************函数体外*********************************#
print('\n下雪了……\n')
print('=============== 开始做梦…… ==============\n')
fun_christmastree()                                    # 调用函数
print('=============== 梦醒了…… ===============\n')
pinetree = '我身上落满雪花，' + pinetree + ' -_- '        # 为全局变量赋值
print(pinetree)                                        # 输出全局变量的值
```

在与 christmastree 模块同级的目录下，创建一个名称为 main.py 的文件，在该文件中，导入 christmastree 模块，再通过 print()语句输出模块中的全局变量 pinetree 的值，代码如下：

```
import differenttree                          # 导入 differenttree 模块
print(differenttree.pinetree)
```

执行上面的代码，将显示如图 10.6 所示的内容。

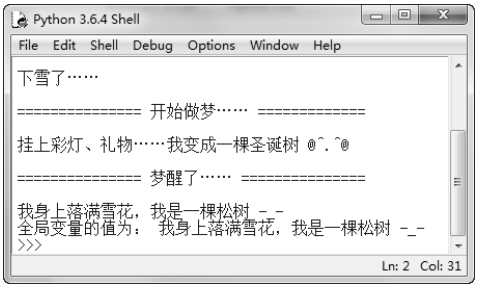


图 10.6　导入输出模块中定义的全局变量的值

从图 10.6 所示的运行结果中可以看出，导入模块后，不仅仅输出了全局变量的值，而且模块中原有的测试代码也被执行了。这个结果显然不是我们想要的，那么如何只输出全局变量的值呢？实际上，可以在模块中，将原本直接执行的测试代码放在一个 if 语句中。因此，可以将模块 christmastree 的代码修改为以下内容：

```
pinetree = '我是一棵松树'  # 定义一个全局变量（松树）
def fun_christmastree():  # 定义函数
    '''功能：一个梦
       无返回值
    '''
    pinetree = '挂上彩灯、礼物……我变成一棵圣诞树 @^.^@ \n'  # 定义局部变量赋值
    print(pinetree)  # 输出局部变量的值
# *************************判断是否以主程序的形式运行**************************#
if __name__ == '__main__':
    print('\n下雪了……\n')
    print('=============== 开始做梦…… ==============\n')
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
12      fun_christmastree()  # 调用函数
13      print('=============== 梦醒了…… ===============\n')
14      pinetree = '我身上落满雪花，' + pinetree + ' -_- '  # 为全局变量赋值
15      print(pinetree)  # 输出全局变量的值
```

再次执行导入模块的 main.py 文件，将显示如图 10.7 所示的结果。从执行结果中可以看出，测试代码并没有执行。

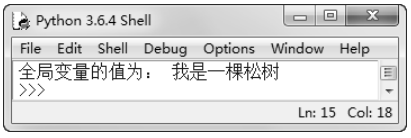


图 10.7　在模块中加入以主程序的形式执行的判断

此时，如果执行 christmastree.py 文件，将显示如图 10.8 所示的结果。

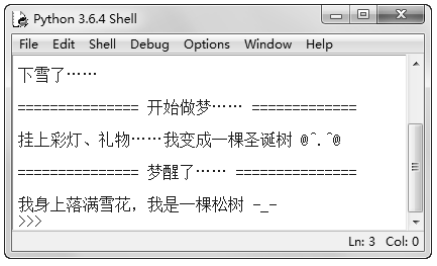


图 10.8　以主程序的形式执行的结果

> **说明：**在每个模块的定义中都包括一个记录模块名称的变量__name__，程序可以检查该变量，以确定它们在哪个模块中执行。如果一个模块不是被导入其他程序中执行的，那么它可能在解释器的顶级模块中执行。顶级模块的__name__变量的值为__main__。

10.4　Python中的包

使用模块可以避免函数名和变量名重名引发的冲突。那么，如果模块名重复应该怎么办呢？在 Python 中，提出了包（Package）的概念。包是一个分层次的目录结构，它将一组功能相近的模块组织在一个目录下。这样，既可以起到规范代码的作用，又能避免模块名重名引起的冲突。

> **说明：**包简单理解就是“文件夹”，只不过在该文件夹下必须存在一个名称为“__init__.py”的文件。

10.4.1　Python程序的包结构

在实际项目开发时，在通常情况下，会创建多个包用于存放不同类的文件。例如，开发一个网站时，可以创建如图 10.9 所示的包结构。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

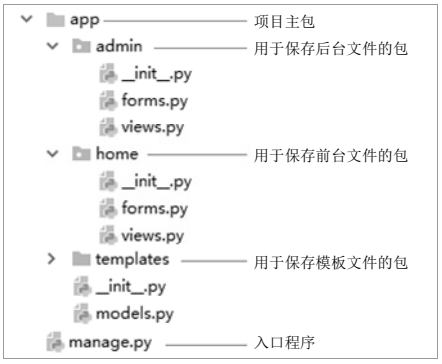


图 10.9　一个 Python 项目的包结构

> **说明：**在图 10.9 中，先创建一个名称为 app 的包，然后在该包下又创建了 admin、home 和 templates 3 个子包和一个 manage.py 的文件，最后在每个子包中，又创建了相应的模块。

10.4.2　创建和使用包

下面将分别介绍如何创建和使用包。

1．创建包

创建包实际上就是创建一个文件夹，并且在该文件夹中创建一个名称为“__init__.py”的 Python 文件。在__init__.py 文件中，可以不编写任何代码，也可以编写一些 Python 代码。在__init__.py 文件中所编写的代码，在导入包时会自动执行。

> **说明：**__init__.py 文件是一个模块文件，模块名为对应的包名。例如，在 settings 包中创建的__init__.py 文件，对应的模块名为 settings。

例如，在 E 盘根目录下创建一个名称为 settings 的包，可以按照以下步骤进行。

（1）在计算机的 E 盘根目录下，创建一个名称为 settings 的文件夹。

（2）在 IDLE 中，创建一个名称为“__init__.py”的文件，保存在 E:\settings 文件夹下，并且在该文件中不写任何内容，然后再返回到资源管理器中，效果如图 10.10 所示。

图 10.10　创建__init__.py 文件后的效果

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

至此，名称为 settings 的包就创建完毕了。之后就可以在该包中创建所需的模块了。

2．使用包

创建包以后，就可以在包中创建相应的模块，然后再使用 import 语句从包中加载模块。从包中加载模块通常有以下 3 种方式。

（1）通过“import+完整包名+模块名”的方式加载指定模块。

“import+完整包名+模块名”的方式是指：假如有一个名称为 settings 的包，在该包下有一个名称为 size 的模块，那么要导入 size 模块，可以使用下面的代码：

```
import settings.size
```

通过该方式导入模块后，在使用时需要使用完整的名称。例如，在已经创建的 settings 包中创建一个名称为 size 的模块，并且在该模块中定义两个变量，代码如下：

```
01 width = 800                # 宽度
02 height = 600               # 高度
```

这时，通过“import+完整包名+模块名”的方式导入 size 模块后，在调用 width 和 height 变量时，就需要在变量名前加入“settings.size.”前缀。对应的代码如下：

```
01 import settings.size       # 导入 settings 包下的 size 模块
02 if __name__=='__main__':
03     print('宽度: ',settings.size.width)
04     print('高度: ',settings.size.height)
```

执行上面的代码后，将显示以下内容：

```
宽度:  800
高度:  600
```

（2）通过“from+完整包名+import+模块名”的方式加载指定模块。

“from+完整包名+import+模块名”的方式是指：假如有一个名称为 settings 的包，在该包下有一个名称为 size 的模块，那么要导入 size 模块，可以使用下面的代码：

```
from settings import size
```

通过该方式导入模块后，在使用时不需要带包前缀，但是需要带模块名。例如，想通过“from+完整包名+import+模块名”的方式导入上面已经创建的 size 模块，并且调用 width 和 height 变量，就可以通过下面的代码实现：

```
01 from settings import size    # 导入 settings 包下的 size 模块
02 if __name__=='__main__':
03     print('宽度: ',size.width)
04     print('高度: ',size.height)
```

执行上面的代码后，将显示以下内容：

```
宽度:  800
高度:  600
```

（3）通过“from+完整包名+import+模块名+定义名”的方式加载指定模块。

“from+完整包名+import+模块名+定义名”的方式是指：假如有一个名称为 settings 的包，

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

在该包下有一个名称为 size 的模块，那么要导入 size 模块中的 width 和 height 变量，可以使用下面的代码：

```
from settings.size import width,height
```

通过该方式导入模块的函数、变量或类后，在使用时直接使用函数、变量或类名即可。例如，想通过“from+完整包名+import+模块名+定义名”的方式导入上面已经创建的 size 模块的 width 和 height 变量，并输出，就可以通过下面的代码实现：

```
# 导入 settings 包下 size 模块中的 width 和 height 变量
from settings.size import width,height
if __name__=='__main__':
    print('宽度：', width)    # 输出宽度
    print('高度：', height)   # 输出高度
```

执行上面的代码后，将显示以下内容：

```
宽度： 800
高度： 600
```

> **说明：**在通过“from+完整包名+import+模块名+定义名”的方式加载指定模块时，可以使用星号“*”代替定义名，表示加载该模块下的全部定义。

10.5 引用其他模块

在 Python 中，除了可以自定义模块外，还可以引用其他模块，主要包括使用标准模块和第三方模块。下面分别进行介绍。

在 Python 中，自带了很多实用的模块，称为标准模块（也可以称为标准库），对于标准模块，我们可以直接使用 import 语句导入 Python 文件中使用。例如，导入标准模块 random（用于生成随机数），可以使用下面的代码：

```
import random                          # 导入标准模块 random
```

> **说明：**在导入标准模块时，也可以使用 as 关键字为其指定别名。在通常情况下，如果模块名比较长，则可以为其设置别名。

导入标准模块后，可以通过模块名调用其提供的函数。例如，导入 random 模块后，就可以调用其 randint()函数生成一个指定范围的随机整数。生成一个 0～10 之间（包括 0 和 10）的随机整数的代码如下：

```
import random                          # 导入标准模块 random
print(random.randint(0,10))    # 输出 0~10 之间的随机整数
```

执行上面的代码，可能会输出 0～10 之间的任意一个数。

除了 random 模块外，Python 还提供了大约 200 多个内置的标准模块，涵盖了 Python 运行时服务、文字模式匹配、操作系统接口、数学运算、对象永久保存、网络和 Internet 脚本，以

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

及 GUI 建构等方面。Python 常用的内置标准模块如表 10.1 所示。

表 10.1　Python 常用的内置标准模块

模 块 名	描　　述
sys	与 Python 解释器及其环境操作相关的标准库
time	提供与时间相关的各种函数的标准库
os	提供了访问操作系统服务功能的标准库
calendar	提供与日期相关的各种函数的标准库
urllib	用于读取来自网上（服务器上）的数据的标准库
json	用于使用 JSON 序列化和反序列化对象
re	用于在字符串中执行正则表达式匹配和替换
math	提供标准算术运算函数的标准库
decimal	用于进行精确控制运算精度、有效数位和四舍五入操作的十进制运算
shutil	用于进行高级文件操作，如复制、移动和重命名等
logging	提供了灵活的记录事件、错误、警告和调试信息等日志信息的功能
tkinter	使用 Python 进行 GUI 编程的标准库

除了表 10.1 所列出的标准模块外，Python 还提供了很多标准模块，读者可以在 Python 的帮助文档中查看。具体方法是：打开 Python 安装路径下的 Doc 目录，在该目录中的扩展名为.chm 的文件（如 python364.chm）即为 Python 的帮助文档。打开该文件，找到如图 10.11 所示的位置进行查看即可。

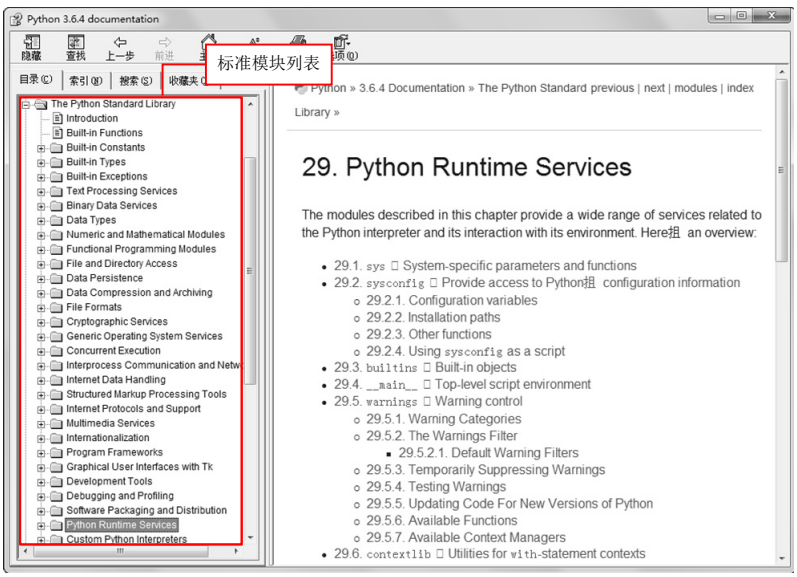


图 10.11　Python 的帮助文档

10.6　快学快用

（1）验证码（CAPTCHA）是 Completely Automated Public Turing test to tell Computers and

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

Humans Apart（全自动区分计算机和人类的图灵测试）的缩写，是一种区分用户是计算机还是人的公共全自动程序。如铁路售票官网 12306 推出的图片验证码，就是防止机器刷票的验证码程序，如图 10.12 所示。写一个 6 位随机验证码程序，使用 random 模块，要求验证码中至少包含一个数字、一个小写字母、一个大写字母。

图 10.12　图片验证

程序参考代码如下：

```
import randomimport string

code_li = []

code_li.append(random.choice(string.ascii_lowercase))
code_li.append(random.choice(string.digits))
code_li.append(random.choice(string.ascii_uppercase))while len(code_li) < 6:
code_li.append(random.choice(string.digits+string.ascii_lowercase+string.
ascii_uppercase))print(code_li)
q_code=''.join(code_li)print(q_code)
```

（2）在应用程序的开发过程中，难免要与日期、时间处理打交道。比如，记录一个复杂算法的执行时间，将时间 2019-02-05 20:28:54 转换成时间戳。

参考代码如下：

```
 #coding:UTF-8
import time

dt = "2019-02-05 20:28:54"
#转换成时间数组
timeArray = time.strptime(dt, "%Y-%m-%d %H:%M:%S")     #转换成时间戳
timestamp = time.mktime(timeArray)
print timestamp
```

（3）Python 提供了很多有用的内置函数或模块，大家熟悉一下这些函数或模块的用途，在实际开发时如果遇到相关应用，可以联想到相关函数或模块，然后详细学习并使用，事半功倍。如表 10.2 所示为反射内置函数。如果要详细了解这些函数或模块的使用，可以扫描二维

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

码，到快用云端支持平台学习、训练、实战。

表 10.2　反射内置函数(2)

eval(expression [, globals [, locals]])	计算表达式 expression 的值
execfile(filename [, globals [, locals]])	用法类似 exec()，不同的是 execfile 的参数 filename 为文件名，而 exec 的参数为字符串
filter(function, iterable)	构造一个序列，等价于[item for item in iterable if function(item)] • 参数 function：返回值为 True 或 False 的函数，可以为 None • 参数 iterable：序列或可迭代对象
getattr(object, name [, defalut])	获取一个类的属性
globals()	返回一个描述当前全局符号表的字典
hasattr(object, name)	判断对象 object 是否包含名为 name 的特性
hash(object)	如果对象 object 为哈希表类型，返回对象 object 的哈希值
id(object)	返回对象的唯一标识
isinstance(object, classinfo)	判断 object 是否是 class 的实例
issubclass(class, classinfo)	判断是否是子类
len(s)	返回集合长度

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 11 章

文件及目录操作

在变量、序列和对象中，数据是暂时存储的，程序结束后就会丢失。为了能够长时间保存程序中的数据，需要将程序中的数据保存到磁盘文件中。Python 提供了内置的文件对象和对文件、目录进行操作的内置模块。通过这些技术可以很方便地将数据保存到文件（如文本文件等）中，以达到长时间保存数据的目的。

本章将详细介绍在 Python 中如何进行文件和目录的相关操作。

11.1 基本文件操作

在 Python 中，内置了文件（File）对象。在使用文件对象时，首先需要通过内置的 open()函数创建一个文件对象，然后通过该对象提供的方法进行一些基本的文件操作。例如，可以使用文件对象的 write()方法向文件中写内容，以及使用 close()方法关闭文件等。下面将介绍如何应用 Python 的文件对象进行基本文件操作。

11.1.1 创建和打开文件

在 Python 中，想要操作文件，需要先创建或者打开指定的文件并创建文件对象，可以通过内置的 open()函数实现。open()函数的基本语法格式如下：

```
file = open(filename[,mode[,buffering]])
```

参数说明如下。

- file：被创建的文件对象。
- filename：要创建或打开文件的文件名称，需要使用单引号或双引号括起来。如果要打开的文件和当前文件在同一个目录下，那么直接写文件名即可，否则需要指定完整路径。例如，要打开当前路径下名称为 status.txt 的文件，可以使用"status.txt"。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- mode：可选参数，用于指定文件的打开模式，其参数值如表 11.1 所示。默认的打开模式为只读，即 r。

表 11.1　mode 参数的参数值说明

值	说　明	注　意
r	以只读模式打开文件。文件的指针将会放在文件的开头	文件必须存在
rb	以二进制格式打开文件，并且采用只读模式。文件的指针将会放在文件的开头。一般用于非文本文件，如图片、声音等	
r+	打开文件后，可以读取文件内容，也可以写入新的内容覆盖原有内容（从文件开头进行覆盖）	
rb+	以二进制格式打开文件，并且采用读写模式。文件的指针将会放在文件的开头。一般用于非文本文件，如图片、声音等	
w	以只写模式打开文件	文件存在，则将其覆盖，否则创建新文件
wb	以二进制格式打开文件，并且采用只写模式。一般用于非文本文件，如图片、声音等	
w+	打开文件后，清空原有内容，使其变为一个空的文件，对这个空文件有读写权限	
wb+	以二进制格式打开文件，并且采用读写模式。一般用于非文本文件，如图片、声音等	
a	以追加模式打开一个文件。如果该文件已经存在，文件指针将放在文件的末尾（新内容会被写入已有内容之后），否则，创建新文件用于写入	
ab	以二进制格式打开文件，并且采用追加模式。如果该文件已经存在，文件指针将放在文件的末尾（新内容会被写入已有内容之后），否则，创建新文件用于写入	
a+	以读写模式打开文件。如果该文件已经存在，文件指针将放在文件的末尾（新内容会被写入已有内容之后），否则，创建新文件用于读写	
ab+	以二进制格式打开文件，并且采用追加模式。如果该文件已经存在，文件指针将放在文件的末尾（新内容会被写入已有内容之后），否则，创建新文件用于读写	

- buffering：可选参数，用于指定读写文件的缓冲模式，值为 0 表示不缓存；值为 1 表示缓存；如果大于 1，则表示缓冲区的大小。默认为缓存模式。

使用 open()函数可以实现以下几个功能。

1．打开一个不存在的文件时先创建该文件

在默认情况下，使用 open()函数打开一个不存在的文件，会抛出如图 11.1 所示的异常。

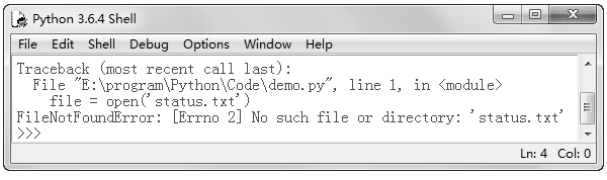


图 11.1　打开的文件不存在时抛出的异常

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

要解决如图 11.1 所示的错误，主要有以下两种方法。

- 在当前目录（与执行的文件相同的目录）下创建一个名称为 status.txt 的文件。
- 在调用 open()函数时，指定 mode 的参数值为 w、w+、a、a+。这样，当要打开的文件不存在时，就可以创建新的文件了。

2．以二进制形式打开文件

使用 open()函数不仅可以以文本形式打开文本文件，还可以以二进制形式打开非文本文件，如图片文件、音频文件、视频文件等。例如，创建一个名称为 picture.png 的图片文件，如图 11.2 所示，并且应用 open()函数以二进制形式打开该文件。

以二进制形式打开该文件，并输出创建的对象，代码如下：

```
01  file = open('picture.png','rb')     # 以二进制形式打开图片文件
02  print(file)                          # 输出创建的对象
```

执行上面的代码后，将显示如图 11.3 所示的运行结果。

图 11.2　打开的图片文件

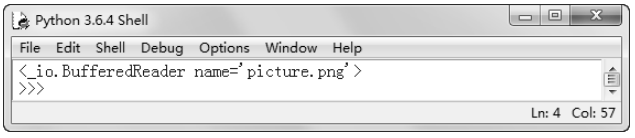


图 11.3　以二进制形式打开图片文件

从图 11.3 中可以看出，创建的是一个 BufferedReader 对象。该对象生成后，可以再应用第三方模块进行处理。例如，上面的 BufferedReader 对象是通过打开图片文件实现的，那么就可以将其传入第三方的图像处理库 PIL 的 Image 模块的 open 方法中，以便对图片进行处理（如调整大小等）。

3．打开文件时指定编码方式

在使用 open()函数打开文件时，默认采用 GBK 编码，当被打开的文件不是 GBK 编码时，将抛出如图 11.4 所示的异常。

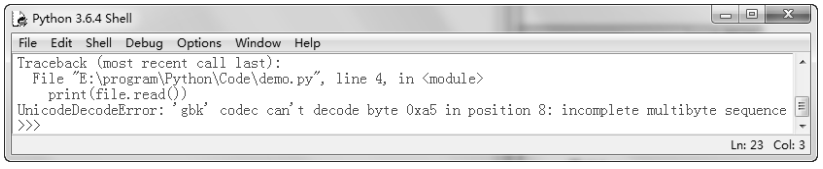


图 11.4　抛出 Unicode 解码异常

解决该问题的方法有两种，一种是直接修改文件的编码，另一种是在打开文件时，直接指定使用的编码方式。推荐采用后一种方法。下面重点介绍如何在打开文件时指定编码方式。

在调用 open()函数时，通过添加 encoding='utf-8'参数即可将编码指定为 UTF-8。如果想要指定其他编码，可以将单引号中的内容替换为想要指定的编码。

例如，打开采用 UTF-8 编码保存的 notice.txt 文件，可以使用下面的代码：

```
file = open('notice.txt','r',encoding='utf-8')
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

11.1.2　关闭文件

打开文件后，需要及时关闭，以免对文件造成不必要的破坏。关闭文件可以使用文件对象的 close()方法实现。close()方法的语法格式如下：

```
file.close()
```

其中，file 为打开的文件对象。

例如，关闭打开的 file 对象，可以使用下面的代码：

```
file.close()    # 关闭文件对象
```

> **说明：**close()方法先刷新缓冲区中还没有写入的信息，再关闭文件，这样可以将没有写入文件的内容写入文件。在关闭文件后，便不能再进行写入操作了。

11.1.3　打开文件时使用with语句

打开文件后，要及时将其关闭，如果忘记关闭可能会带来意想不到的问题。另外，如果在打开文件时抛出了异常，那么将导致文件不能被及时关闭。为了更好地避免此类问题发生，可以使用 Python 提供的 with 语句，从而实现在处理文件时，无论是否抛出异常，都能保证 with 语句执行完毕后关闭已经打开的文件。with 语句的基本语法格式如下：

```
with expression as target:
    with-body
```

参数说明如下。

- expression：用于指定一个表达式，这里可以是打开文件的 open()函数。
- target：用于指定一个变量，并且将 expression 的结果保存到该变量中。
- with-body：用于指定 with 语句体，其中可以是执行 with 语句后相关的一些操作语句。如果不想执行任何语句，可以直接使用 pass 语句代替。

例如，在打开文件时使用 with 语句，修改后的代码如下：

```
01 print("\n","="*10,"蚂蚁庄园动态","="*10)
02 with open('message.txt','w') as file:   # 创建或打开保存蚂蚁庄园动态信息的文件
03     pass
04 print("\n 即将显示……\n")
```

11.1.4　写入文件内容

前面的示例中虽然创建并打开了一个文件，但是该文件中并没有任何内容，它的大小是 0KB。Python 的文件对象提供了 write()方法，可以向文件中写入内容。write()方法的语法格式如下：

```
file.write(string)
```

其中，file 为打开的文件对象；string 为要写入的字符串。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **注意：**调用 write()方法向文件中写入内容的前提是，在打开文件时，指定的打开模式为 w（可写）或者 a（追加），否则，将抛出如图 11.5 所示的异常。

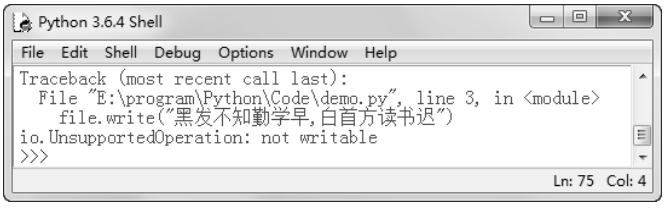


图 11.5　没有写入权限时抛出的异常

11.1.5　读取文件

在 Python 中打开文件后，除了可以向其写入或追加内容外，还可以读取文件中的内容。读取文件内容主要分为以下几种情况。

1．读取指定字符

文件对象提供了 read()方法读取指定个数的字符，其语法格式如下：

```
file.read([size])
```

其中，file 为打开的文件对象；size 为可选参数，用于指定要读取的字符个数，如果省略，则一次性读取所有内容。

> **注意：**调用 read()方法读取文件内容的前提是，在打开文件时，指定的打开模式为 r（只读）或者 r+（读写），否则，将抛出如图 11.6 所示的异常。

图 11.6　没有读取权限时抛出的异常

例如，要读取 message.txt 文件中的前 9 个字符，可以使用下面的代码：

```
01 with open('message.txt','r') as file:  # 打开文件
02     string = file.read(9)              # 读取前 9 个字符
03     print(string)
```

如果 message.txt 的文件内容为：

```
你使用了 1 张加速卡，小鸡撸起袖子开始双手吃饲料，进食速度大大加快。
```

那么执行上面的代码将显示以下结果：

```
你使用了 1 张加速卡
```

使用 read(size)方法读取文件时，是从文件的开头读取的。如果想要读取部分内容，可以

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

先使用文件对象的 seek()方法将文件的指针移动到新的位置，再应用 read(size)方法读取。seek()方法的基本语法格式如下：

```
file.seek(offset[,whence])
```

参数说明如下。

- file：表示已经打开的文件对象。
- offset：用于指定移动的字符个数，其具体位置与 whence 有关。
- whence：用于指定从什么位置开始计算。值 0 表示从文件头开始计算，1 表示从当前位置开始计算，2 表示从文件尾开始计算，默认为 0。

> **注意：**对于 whence 参数，如果在打开文件时，没有使用 b 模式（rb），那么只允许从文件头开始计算相对位置，从文件尾计算时就会引发如图 11.7 所示的异常。

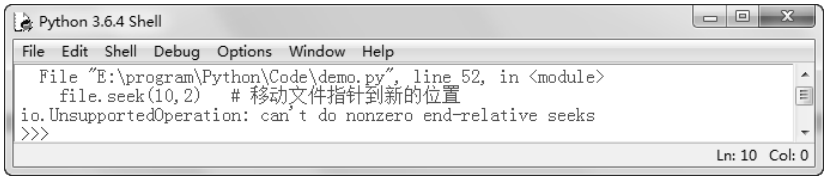


图 11.7　抛出 io.UnsupportedOperation 异常

例如，想要从文件的第 19 个字符开始读取 13 个字符，可以使用下面的代码。

```
with open('message.txt','r') as file:  # 打开文件
    file.seek(19)                      # 移动文件指针到新的位置
    string = file.read(13)             # 读取 13 个字符
    print(string)
```

如果 message.txt 的文件内容为：

```
你使用了 1 张加速卡，小鸡撸起袖子开始双手吃饲料，进食速度大大加快。
```

那么执行上面的代码将显示以下结果：

```
小鸡撸起袖子开始双手吃饲料
```

> **说明：**在使用 seek()方法时，offset 的值是按一个汉字占两个字符、一个英文字母和数字占一个字符计算的。这与 read(size)方法不同。

2．读取一行

在使用 read()方法读取文件时，如果文件很大，一次读取全部内容到内存，容易造成内存不足，所以通常会采用逐行读取。文件对象提供了 readline()方法用于每次读取一行数据。readline()方法的基本语法格式如下：

```
file.readline()
```

其中，file 为打开的文件对象。同 read()方法一样，在打开文件时，也需要指定打开模式为 r（只读）或者 r+（读写）。

```
file = open('message.txt','w')          # 创建或打开保存蚂蚁庄园动态信息的文件
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
print("\n 即将显示……\n")
print("\n","="*35,"蚂蚁庄园动态","="*35,"\n")
with open('message.txt','r') as file:   # 打开保存蚂蚁庄园动态信息的文件
    number = 0   # 记录行号
    while True:
        number += 1
        line = file.readline()
        if line =='':
            break    # 跳出循环
        print(number,line,end= "\n")  # 输出一行内容
print("\n","="*39,"over","="*39,"\n")
```

执行上面的代码，将显示如图 11.8 所示的结果。

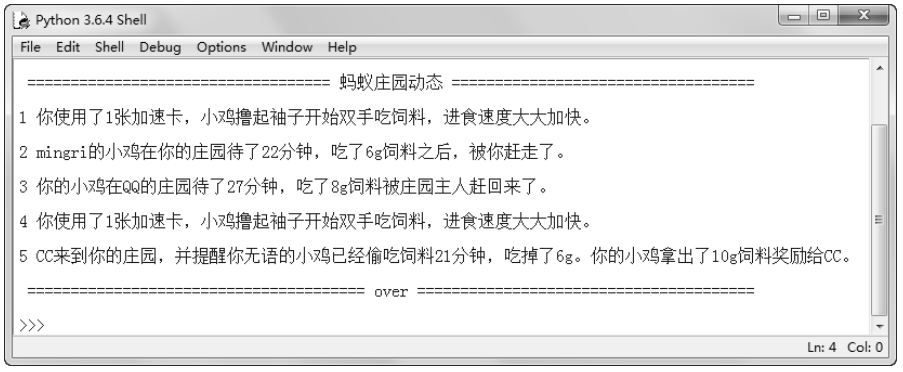


图 11.8　逐行显示蚂蚁庄园的全部动态

3．读取全部行

读取全部行的作用同调用 read()方法时不指定 size 类似，只不过读取全部行时，返回的是一个字符串列表，每个元素为文件的一行内容。读取全部行，使用的是文件对象的 readlines()方法，其语法格式如下：

```
file.readlines()
```

其中，file 为打开的文件对象。同 read()方法一样，在打开文件时，也需要指定打开模式为 r（只读）或者 r+（读写）。

例如，通过 readlines()方法读取 message.txt 文件，并输出读取结果，代码如下：

```
print("\n","="*25,"蚂蚁庄园动态","="*25,"\n")
with open('message.txt','r') as file:      # 打开保存蚂蚁庄园动态信息的文件
    message = file.readlines()             # 读取全部动态信息
    print(message)                         # 输出动态信息
    print("\n","="*29,"over","="*29,"\n")
```

执行上面的代码，将显示如图 11.9 所示的运行结果。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

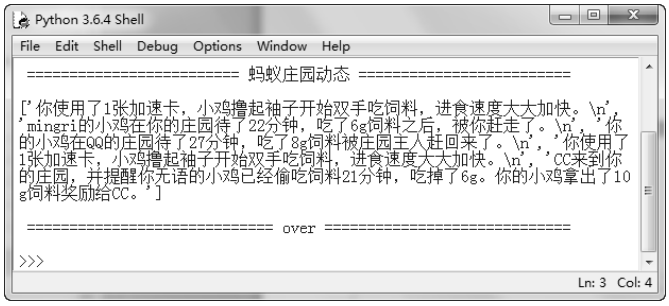


图 11.9　readlines()方法的返回结果

从运行结果中可以看出，readlines()方法的返回值为一个字符串列表，在这个字符串列表中，每个元素记录一行内容。如果文件比较大时，采用这种方法输出读取的文件内容会很慢，这时可以将列表的内容逐行输出。例如，上面的代码可以修改为以下内容。

```
01 print("\n","="*25,"蚂蚁庄园动态","="*25,"\n")
02 with open('message.txt','r') as file:          # 打开保存蚂蚁庄园动态信息的文件
03     messageall = file.readlines()              # 读取全部动态信息
04     for message in messageall:
05         print(message)                         # 输出一条动态信息
06 print("\n","="*29,"over","="*29,"\n")
```

执行结果如图 11.10 所示。

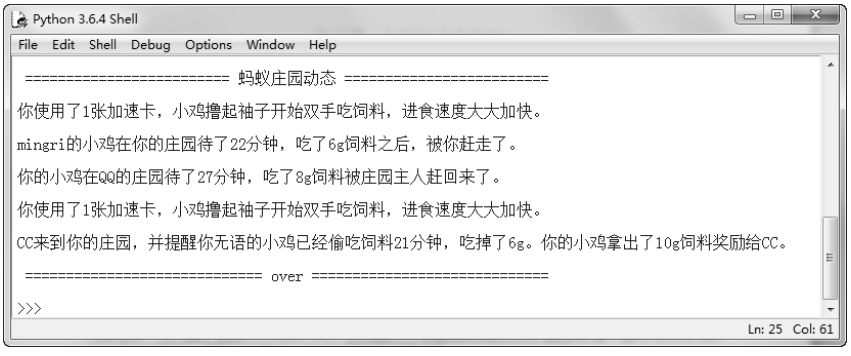


图 11.10　应用 readlines()方法逐行输出动态信息

11.2　目录操作

目录也称文件夹，用于分层保存文件。通过目录可以分门别类地存放文件，我们也可以通过目录快速找到想要的文件。在 Python 中，并没有提供直接操作目录的函数或者对象，而是需要使用内置的 os 和 os.path 模块实现。

> **说明：**os 模块是 Python 内置的与操作系统功能和文件系统相关的模块。该模块中的语句的执行结果通常与操作系统有关，在不同的操作系统上运行，可能会得到不一样的结果。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

常用的目录操作主要有判断目录是否存在、创建目录、删除目录和遍历目录等，下面将进行详细介绍。

> **说明：**本章的内容都是以 Windows 操作系统为例进行介绍的，所以代码的执行结果也都是在 Windows 操作系统下显示的。

11.2.1 os和os.path模块

在 Python 中，内置了 os 模块及其子模块 os.path 用于对目录或文件进行操作。在使用 os 模块或者 os.path 模块时，需要先应用 import 语句将其导入，然后才可以应用它们提供的函数或者变量。

可以使用下面的代码导入 os 模块：

```
import os
```

> **说明：**导入 os 模块后，也可以使用其子模块 os.path。

导入 os 模块后，可以使用该模块提供的通用变量获取与系统有关的信息。常用的变量有以下几个。

（1）name：用于获取操作系统类型。

例如，在 Windows 操作系统下输出 os.name，将显示如图 11.11 所示的结果。

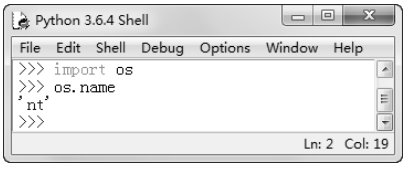


图 11.11　显示 os.name 的结果

> **说明：**如果 os.name 的输出结果为 nt，则表示是 Windows 操作系统；如果是 posix，则表示是 Linux、UNIX 或 Mac OS 操作系统。

（2）linesep：用于获取当前操作系统上的换行符。

例如，在 Windows 操作系统下输出 os.linesep，将显示如图 11.12 所示的结果。

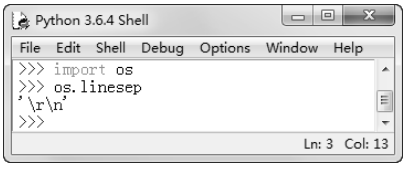


图 11.12　显示 os.linesep 的结果

（3）sep：用于获取当前操作系统所使用的路径分隔符。

例如，在 Windows 操作系统下输出 os.sep，将显示如图 11.13 所示的结果。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

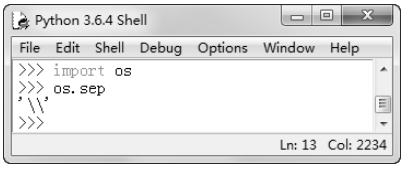


图 11.13　显示 os.sep 的结果

os 模块还提供了一些操作目录的函数，如表 11.2 所示。

表 11.2　os 模块提供的与目录相关的函数

函　　数	说　　明
getcwd()	返回当前的工作目录
listdir(path)	返回指定路径下的文件和目录信息
mkdir(path [,mode])	创建目录
makedirs(path1/path2……[,mode])	创建多级目录
rmdir(path)	删除目录
removedirs(path1/path2……)	删除多级目录
chdir（path）	把 path 设置为当前工作目录
walk(top[,topdown[,onerror]])	遍历目录树，该方法返回一个元组，包括所有路径名、所有目录列表和文件列表 3 个元素

os.path 模块也提供了一些操作目录的函数，如表 11.3 所示。

表 11.3　os.path 模块提供的与目录相关的函数

函　　数	说　　明
abspath(path)	用于获取文件或目录的绝对路径
exists(path)	用于判断目录或者文件是否存在，如果存在则返回 True，否则返回 False
join(path,name)	将目录与目录或者文件名拼接起来
splitext()	分离文件名和扩展名
basename(path)	从一个目录中提取文件名
dirname(path)	从一个路径中提取文件路径，不包括文件名
isdir(path)	用于判断是否为路径

11.2.2　路径

用于定位一个文件或者目录的字符串被称为一个路径。在程序开发中，通常涉及两种路径，一种是相对路径，另一种是绝对路径。

1．相对路径

在学习相对路径之前，需要了解什么是当前工作目录。当前工作目录是指当前文件所在的目录。在 Python 中，可以通过 os 模块提供的 getcwd()函数获取当前工作目录。例如，在 E:\program\Python\Code\demo.py 文件中，编写以下代码：

```
01 import os
02 print(os.getcwd())    # 输出当前目录
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

执行上面的代码后，将显示以下目录，该路径就是当前工作目录。

```
E:\program\Python\Code
```

相对路径就是依赖于当前工作目录的。如果在当前工作目录下，有一个名称为 message.txt 的文件，那么在打开这个文件时，就可以直接写上文件名，这时采用的就是相对路径，message.txt 文件的实际路径就是当前工作目录“E:\program\Python\Code”+相对路径“message.txt”，即“E:\program\Python\Code\message.txt”。

如果在当前工作目录下，有一个子目录 demo，并且在该子目录下保存着文件 message.txt，那么在打开这个文件时就可以写上“demo/message.txt”，例如下面的代码：

```
01 with open("demo/message.txt") as file:        # 通过相对路径打开文件
02     pass
```

> **说明：**在 Python 中，指定文件路径时需要对路径分隔符“\”进行转义，即将路径中的“\”替换为“\\”。例如对于相对路径“demo\message.txt”需要使用“demo\\message.txt”代替。另外，也可以将路径分隔符“\”采用“/”代替。

多学两招

在指定文件路径时，也可以在表示路径的字符串前面加上字母 r（或 R），那么该字符串将原样输出，这时路径中的分隔符就不需要再转义了。例如，上面的代码也可以修改为如下内容：

```
01 with open(r"demo\message.txt") as file:        # 通过相对路径打开文件
02     pass
```

2．绝对路径

绝对路径是指在使用文件时指定文件的实际路径，它不依赖于当前工作目录。在 Python 中，可以通过 os.path 模块提供的 abspath()函数获取一个文件的绝对路径。abspath()函数的基本语法格式如下：

```
os.path.abspath(path)
```

其中，path 为要获取的绝对路径的相对路径，可以是文件，也可以是目录。

例如，要获取相对路径“demo\message.txt”的绝对路径，可以使用下面的代码：

```
01 import os
02 print(os.path.abspath(r"demo\message.txt"))        # 获取绝对路径
```

如果当前工作目录为“E:\program\Python\Code”，那么将得到以下结果：

```
E:\program\Python\Code\demo\message.txt
```

3．拼接路径

如果想要将两个或者多个路径拼接到一起组成一个新的路径，可以使用 os.path 模块提供的 join()函数实现。join()函数的基本语法格式如下：

```
os.path.join(path1[,path2[,……]])
```

其中，path1、path2 代表要拼接的文件路径，这些路径之间使用逗号进行分隔。如果在要拼接的路径中没有一个绝对路径，那么最后拼接出来的将是一个相对路径。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **注意**：使用 os.path.join()函数拼接路径时，并不会检测该路径是否真实存在。

例如，需要将“E:\program\Python\Code”和“demo\message.txt”路径拼接到一起，可以使用下面的代码：

```
01 import os
02 print(os.path.join("E:\program\Python\Code","demo\message.txt")) # 拼接字符串
```

执行上面的代码，将得到以下结果：

```
E:\program\Python\Code\demo\message.txt
```

> **说明**：在使用 join()函数时，如果要拼接的路径中存在多个绝对路径，那么以从左到右最后一次出现的为准，并且该路径之前的参数都将被忽略。例如，执行下面的代码：
>
> ```
> 01   import os
> 02
>  print(os.path.join("E:\\code","E:\\python\\mr","Code","C:\\","demo"))
> # 拼接字符串
> ```
>
> 将得到拼接后的路径“C:\demo”。

> **注意**：当把两个路径拼接为一个路径时，不要直接使用字符串拼接，而是使用 os.path.join()函数，这样可以正确处理不同操作系统的路径分隔符。

11.2.3　判断目录是否存在

在 Python 中，有时需要判断给定的目录是否存在，这时可以使用 os.path 模块提供的 exists()函数实现。exists()函数的基本语法格式如下：

```
os.path.exists(path)
```

其中，path 为要判断的目录，可以采用绝对路径，也可以采用相对路径。

返回值：如果给定的路径存在，则返回 True，否则返回 False。

例如，要判断绝对路径“C:\demo”是否存在，可以使用下面的代码：

```
01 import os
02 print(os.path.exists("C:\\demo"))                    # 判断目录是否存在
```

执行上面的代码，如果在 C 盘根目录下没有 demo 子目录，则返回 False，否则返回 True。

> **说明**：os.path.exists()函数除了可以判断目录是否存在，还可以判断文件是否存在。例如，如果将上面代码中的“C:\\demo”替换为“C:\\demo\\test.txt”，则用于判断 C:\demo\test.txt 文件是否存在。

11.2.4　创建目录

在 Python 中，os 模块提供了两个创建目录的函数，一个用于创建一级目录，另一个用于创建多级目录。下面分别进行介绍。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

1. 创建一级目录

创建一级目录是指一次只能创建一级目录。在 Python 中，可以使用 os 模块提供的 mkdir()函数实现，通过该函数只能创建指定路径中的最后一级目录，如果该目录的上一级不存在，则抛出 FileNotFoundError 异常。mkdir()函数的基本语法格式如下：

```
os.mkdir(path, mode=0o777)
```

参数说明如下。

- path：用于指定要创建的目录，可以使用绝对路径，也可以使用相对路径。
- mode：用于指定数值模式，默认值为 0777。该参数在非 UNIX 系统上无效或被忽略。

例如，在 Windows 系统上创建一个 C:\demo 目录，可以使用下面的代码：

```
01 import os
02 os.mkdir("C:\\demo")   # 创建 C:\demo 目录
```

执行上面的代码后，将在 C 盘根目录下创建一个 demo 目录，如图 11.14 所示。

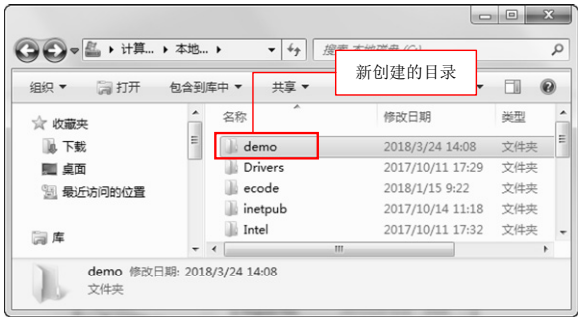


图 11.14　创建 demo 目录成功

如果创建的路径已经存在，将抛出 FileExistsError 异常，例如，将上面的示例代码再执行一次，将抛出如图 11.15 所示的异常。

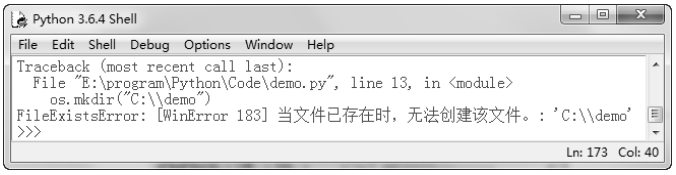


图 11.15　创建 demo 目录失败

要解决上面的问题，可以在创建目录前判断指定的目录是否存在，只有当目录不存在时才创建。具体代码如下：

```
01 import os
02 path = "C:\\demo"                 # 指定要创建的目录
03 if not os.path.exists(path):      # 判断目录是否存在
04     os.makedirs(path)             # 创建目录
05     print("目录创建成功！")
06 else:
07     print("该目录已经存在！")
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

执行上面的代码，将显示“该目录已经存在!”。

> **注意：**如果指定的目录有多级，而且最后一级的上级目录中有不存在的，则抛出 FileNotFoundError 异常，并且目录创建不成功。要解决该问题，可以使用创建多级目录的方法（将在下面进行介绍）。

2. 创建多级目录

使用 mkdir()函数只能创建一级目录。如果想创建多级目录，可以使用 os 模块提供的 makedirs()函数，该函数采用递归的方式创建目录。makedirs()函数的基本语法格式如下：

```
os.makedirs(name, mode=0o777)
```

参数说明如下。

- name：用于指定要创建的目录，可以使用绝对路径，也可以使用相对路径。
- mode：用于指定数值模式，默认值为 0777。该参数在非 UNIX 系统上无效或被忽略。

例如，在 Windows 系统上，在刚刚创建的 C:\demo 目录下再创建子目录 test\dir\mr（对应的目录为 C:\demo\test\dir\mr），可以使用下面的代码：

```
01 import os
02 os. makedirs ("C:\\demo\\test\\dir\\mr ")   # 创建 C:\demo\test\dir\mr 目录
```

执行上面的代码后，将在 C:\demo 目录下创建子目录 test，并且在 test 目录下再创建子目录 dir，在 dir 目录下再创建子目录 mr。创建后的目录结构如图 11.16 所示。

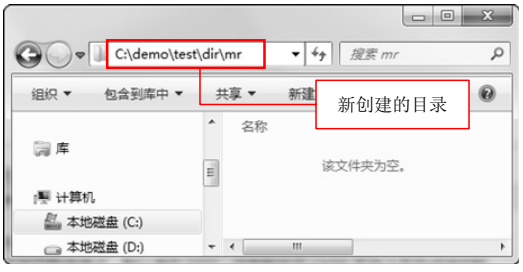


图 11.16　创建多级目录的结果

11.2.5　删除目录

删除目录可以使用 os 模块提供的 rmdir()函数实现。在通过 rmdir()函数删除目录时，只有当要删除的目录为空时才起作用。rmdir()函数的基本语法格式如下：

```
os.rmdir(path)
```

其中，path 为要删除的目录，可以使用相对路径，也可以使用绝对路径。

例如，要删除刚刚创建的 C:\demo\test\dir\mr 目录，可以使用下面的代码：

```
01 import os
02 os.rmdir("C:\\demo\\test\\dir\\mr")   # 删除 C:\demo\test\dir\mr 目录
```

执行上面的代码后，将删除 C:\demo\test\dir 目录下的 mr 目录。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

注意：如果要删除的目录不存在，将抛出“FileNotFoundError: [WinError 2] 系统找不到指定的文件”异常。因此，在执行 os.rmdir()函数前，建议先判断该路径是否存在。可以使用 os.path.exists()函数判断。具体代码如下：

```
01
02  import os
03  path = "C:\\demo\\test\\dir\\mr"              # 指定要创建的目录
04  if os.path.exists(path):                      # 判断目录是否存在
05      os.rmdir("C:\\demo\\test\\dir\\mr")       # 删除目录
06      print("目录删除成功！")
07  else:
08      print("该目录不存在！")
```

多学两招

使用 rmdir()函数只能删除空的目录，如果想要删除非空目录，则需要使用 Python 内置的标准模块 shutil 的 rmtree()函数实现。例如，要删除不为空的 C:\demo\test 目录，可以使用下面的代码：

```
01  import shutil
02  shutil.rmtree("C:\\demo\\test")   # 删除 C:\demo 目录下的 test 子目录及其内容
```

11.2.6 遍历目录

遍历在古汉语中的意思是全部走遍，到处周游。在 Python 中遍历就是对指定的目录下的全部目录（包括子目录）及文件走一遍。在 Python 中，os 模块的 walk()函数用于实现遍历目录的功能。walk()函数的基本语法格式如下：

```
os.walk(top[, topdown][, onerror][, followlinks])
```

参数说明如下。

- top：用于指定要遍历内容的根目录。
- topdown：可选参数，用于指定遍历的顺序。如果值为 True，表示自上而下遍历（先遍历根目录）；如果值为 False，表示自下而上遍历（先遍历最后一级子目录）。默认值为 True。
- onerror：可选参数，用于指定错误处理方式，默认为忽略。如果不想忽略，也可以指定一个错误处理函数。通常情况下采用默认设置。
- followlinks：可选参数，在默认情况下，walk()函数不会向下转换成解析到目录的符号链接。将该参数值设置为 True，表示用于指定在支持的系统上访问由符号链接指向的目录。
- 返回值：返回一个包括 3 个元素（dirpath、dirnames 和 filenames）的元组生成器对象。其中，dirpath 表示当前遍历的路径，是一个字符串；dirnames 表示当前路径下包含的子目录，是一个列表；filenames 表示当前路径下包含的文件，也是一个列表。

例如，要遍历指定目录 E:\program\Python\Code\01，可以使用下面的代码：

```
01  import os                 # 导入 os 模块
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
02 # 遍历 E:\program\Python\Code\01 目录
03 tuples = os.walk("E:\\program\\Python\\Code\\01")
04 for tuple1 in tuples:   # 通过 for 循环输出遍历结果
05     print(tuple1 ,"\n") # 输出每一级目录的元组
```

如果在 E:\program\Python\Code\01 目录下包括如图 11.17 所示的内容，执行上面的代码，将显示如图 11.18 所示的结果。

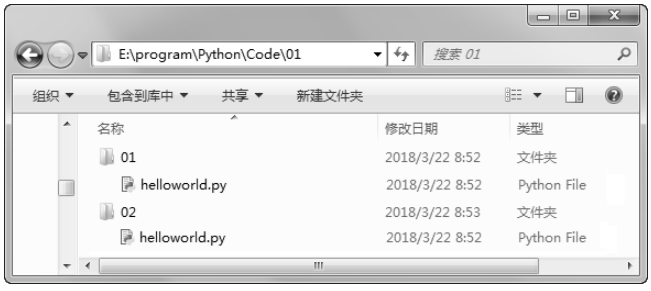


图 11.17　指定目录的内容

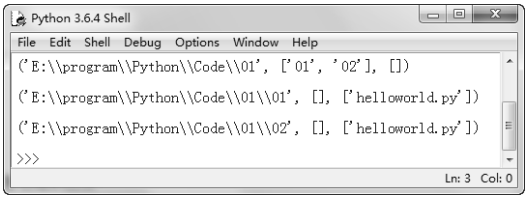


图 11.18　遍历指定目录的结果

注意：walk()函数只在 UNIX 和 Windows 中有效。

11.3　高级文件操作

Python 内置的 os 模块除了可以对目录进行操作外，还可以对文件进行一些高级操作，具体函数如表 11.4 所示。

表 11.4　os 模块提供的与文件相关的函数

函　数	说　明
access(path,accessmode)	获取对文件是否有指定的访问权限（读取/写入/执行权限）。accessmode 的值是 R_OK（读取）、W_OK（写入）、X_OK（执行）或 F_OK（存在）。如果有指定的权限，则返回 1，否则返回 0
chmod(path,mode)	修改 path 指定文件的访问权限
remove(path)	删除 path 指定的文件路径
rename(src,dst)	将文件或目录 src 重命名为 dst
stat(path)	返回 path 指定文件的信息
startfile(path [, operation])	使用关联的应用程序打开 path 指定的文件

下面将对常用的操作进行详细介绍。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

11.3.1 删除文件

Python 没有内置删除文件的函数，但是在内置的 os 模块中提供了删除文件的函数 remove()，该函数的基本语法格式如下：

```
os. remove(path)
```

其中，path 为要删除的文件路径，可以使用相对路径，也可以使用绝对路径。

例如，要删除当前工作目录下的 mrsoft.txt 文件，可以使用下面的代码：

```
01 import os                    # 导入 os 模块
02 os.remove("mrsoft.txt")      # 删除当前工作目录下的 mrsoft.txt 文件
```

执行上面的代码后，如果在当前工作目录下存在 mrsoft.txt 文件，即可将其删除，否则将显示如图 11.19 所示的异常。

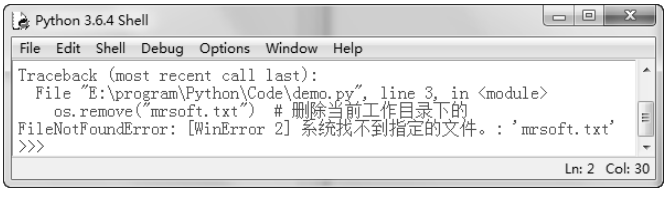


图 11.19 要删除的文件不存在抛出的异常

为了屏蔽以上异常，可以在删除文件时，先判断文件是否存在，只有存在时才执行删除操作。具体代码如下：

```
01 import os                    # 导入 os 模块
02 path = "mrsoft.txt"          # 要删除的文件
03 if os.path.exists(path):     # 判断文件是否存在
04     os.remove(path)          # 删除文件
05     print("文件删除完毕! ")
```

执行上面的代码，如果 mrsoft.txt 文件不存在，则显示以下内容：

```
文件不存在!
```

否则显示以下内容，同时文件被删除：

```
文件删除完毕!
```

11.3.2 重命名文件和目录

os 模块提供了重命名文件和目录的函数 rename()，如果指定的路径是文件，则重命名文件，如果指定的路径是目录，则重命名目录。rename()函数的基本语法格式如下：

```
os.rename(src,dst)
```

其中，src 用于指定要进行重命名的目录或文件；dst 用于指定重命名后的目录或文件。

同删除文件一样，在进行文件或目录重命名时，如果指定的目录或文件不存在，也将抛出 FileNotFoundError 异常，所以在进行文件或目录重命名时，也建议先判断文件或目录是否存在，只有存在时才进行重命名操作。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

例如，想要将“C:\demo\test\dir\mr\mrsoft.txt”文件重命名为“C:\demo\test\dir\mr\mr.txt”，可以使用下面的代码：

```
01 import os                                                  # 导入 os 模块
02 src = "C:\\demo\\test\\dir\\mr\\mrsoft.txt"               # 要重命名的文件
03 dst = "C:\\demo\\test\\dir\\mr\\mr.txt"                   # 重命名后的文件
04 os.rename(src,dst)                                        # 重命名文件
05 if os.path.exists(src):                                   # 判断文件是否存在
06     os.rename(src,dst)                                    # 重命名文件
07     print("文件重命名完毕! ")
08 else:
09     print("文件不存在! ")
```

执行上面的代码，如果 C:\demo\test\dir\mr\mrsoft.txt 文件不存在，则显示以下内容：

```
文件不存在!
```

否则将显示以下内容，同时文件被重命名：

```
文件重命名完毕!
```

使用 rename()函数重命名目录与重命名文件基本相同，只要把原来的文件路径替换为目录即可。例如，想要将当前目录下的 demo 目录重命名为 test，可以使用下面的代码：

```
01 import os                    # 导入 os 模块
02 src = "demo"                 # 要重命名的当前目录下的 demo
03 dst = "test"                 # 目录重命名为 test
04 if os.path.exists(src):      # 判断目录是否存在
05     os.rename(src,dst)       # 重命名目录
06     print("目录重命名完毕! ")
07 else:
08     print("目录不存在! ")
```

> **注意：**在使用 rename()函数重命名目录时，只能修改最后一级的目录名称，否则将抛出如图 11.20 所示的异常。

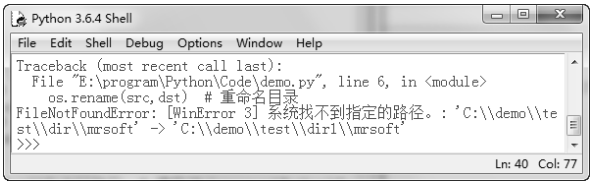


图 11.20　重命名的不是最后一级目录抛出的异常

11.3.3　获取文件基本信息

在计算机上创建文件后，该文件本身就会包含一些信息。例如，文件的最后一次访问时间、最后一次修改时间、文件大小等基本信息。通过 os 模块的 stat()函数可以获取文件的这些基本信息。stat()函数的基本语法如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
os.stat(path)
```

其中，path 为要获取基本信息的文件的路径，可以是相对路径，也可以是绝对路径。

stat()函数的返回值是一个对象，该对象包含如表 11.5 所示的属性。通过访问这些属性可以获取文件的基本信息。

表 11.5　stat()函数返回的对象的常用属性

属　性	说　明	属　性	说　明
st_mode	保护模式	st_dev	设备名
st_ino	索引号	st_uid	用户 ID
st_nlink	硬链接号（被连接数目）	st_gid	组 ID
st_size	文件大小，单位为字节	st_atime	最后一次访问时间
st_mtime	最后一次修改时间	st_ctime	最后一次状态变化的时间（系统不同返回结果也不同，例如，在 Windows 操作系统下返回的是文件的创建时间）

11.4　快学快用

（1）外出旅游，朋友聚会，孩子出生，每次拍上几百张照片。几年下来，电脑里面存储几万张照片也是很常见的事。此时一个突出的问题就会摆在你面前：海量照片如何管理？现在就建立一个照片文件管理小软件，按年份管理你的海量照片（可以先建立一个总的照片文件夹，然后分别根据照片年份建立子文件夹），照片管理效果如图 11.21 所示。

图 11.21　照片管理效果

（2）找出 Windows 文件夹中以“W”开头的文件并输出。输出效果如图 11.22 所示。

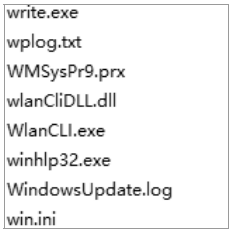


图 11.22　输出效果

（3）复制 Windows 下的日志文件 zh-cn.log 到你的程序目录，修改日志中的所有日期 2018

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

为 2048，并把修改后的文件保存到 2048.txt 中，如图 11.23 和图 11.24 所示。

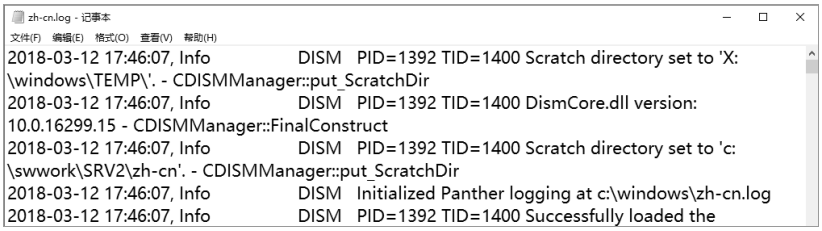


图 11.23　修改前的效果

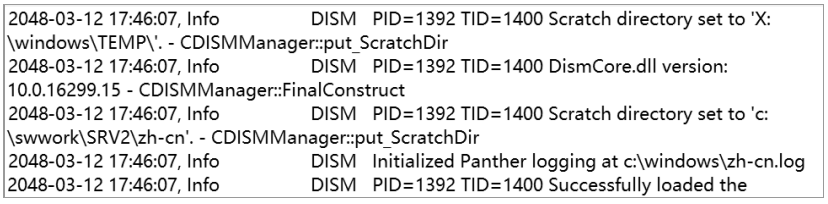


图 11.24　修改后的效果

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 12 章

使用 Python 操作数据库

程序运行的时候，数据都是在内存中的。当程序中止的时候，通常都需要将数据保存到磁盘上，前面我们学习了将数据写入文件，保存在磁盘上。为了便于程序保存和读取数据，并且能直接通过条件快速查询到指定的数据，就出现了数据库（Database）这种专门用于集中存储和查询数据的软件。本章将介绍数据库编程接口的知识，以及使用 SQLite 和 MySQL 存储数据的方法。

12.1 数据库编程接口

在项目开发中，数据库应用必不可少。虽然数据库的种类有很多，如 SQLite、MySQL、Oracle 等，但是它们的功能基本都是一样的，为了对数据库进行统一操作，大多数语言都提供了简单的、标准化的数据库接口（API）。在 Python Database API 2.0 规范中，定义了 Python 数据库 API 接口的各个部分，如模块接口、连接对象、游标对象、类型对象和构造器、DB API 的可选扩展以及可选的错误处理机制等。下面，重点介绍一下数据库 API 接口中的连接对象和游标对象。

12.1.1 连接对象

数据库连接对象（Connection Object）主要提供获取数据库游标对象和提交/回滚事务的方法，以及关闭数据库连接。

1．获取连接对象

如何获取连接对象呢？这就需要使用 connect()函数。该函数有多个参数，具体使用哪个参数，取决于使用的数据库类型。例如，需要访问 Oracle 数据库和 MySQL 数据库，则必须同时下载 Oracle 和 MySQL 数据库模块，这些模块在获取连接对象时，都需要使用 connect()函数。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

connect()函数常用的参数及说明如表 12.1 所示。

表 12.1　connect()函数常用的参数及说明

参　数	说　明
dsn	数据源名称，给出该参数表示数据库依赖
user	用户名
password	用户密码
host	主机名
database	数据库名称

例如，使用 PyMySQL 模块连接 MySQL 数据库，示例代码如下：

```
conn = pymysql.connect(host='localhost',
                       user='user',
                       password='passwd',
                       db='test',
                       charset='utf8',
                       cursorclass=pymysql.cursors.DictCursor)
```

说明：在上述代码中，pymysql.connect()使用的参数与表 12.1 所示的并不是完全相同的。在使用时，要以具体的数据库模块为准。

2．连接对象的方法

Connect()函数返回连接对象，这个对象表示目前和数据库的会话。连接对象支持的方法如表 12.2 所示。

表 12.2　连接对象方法

方 法 名	说　明
close()	关闭数据库连接
commit()	提交事务
rollback()	回滚事务
cursor()	获取游标对象，操作数据库，如执行 DML 操作，调用存储过程等

commit()方法用于提交事务，事务主要用于处理数据量大、复杂度高的数据。如果操作的是一系列的动作，比如张三给李四转账，有如下两个操作：

- 张三账户金额减少。
- 李四账户金额增加。

这时使用事务可以维护数据库的完整性，保证两个操作要么全部执行，要么全部不执行。

12.1.2　游标对象

游标对象（Cursor Object）代表数据库中的游标，用于指示抓取数据操作的上下文，主要提供执行 SQL 语句、调用存储过程、获取查询结果等方法。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

如何获取游标对象呢？通过使用连接对象的 cursor()方法，可以获取游标对象。游标对象的属性如下所示。

- description：数据库列类型和值的描述信息。
- rowcount：返回结果的行数统计信息，如 SELECT、UPDATE、CALLPROC 等。

游标对象的方法如表 12.3 所示。

表 12.3　游标对象的方法

方 法 名	说　明
callproc(procname,[, parameters])	调用存储过程，需要数据库支持
close()	关闭当前游标
execute(operation[, parameters])	执行 SQL 语句或者数据库命令等数据库操作
executemany(operation, seq_of_params)	用于批量操作，如批量更新
fetchone()	获取查询结果集中的下一条记录
fetchmany(size)	获取指定数量的记录
fetchall()	获取结果集的所有记录
nextset()	跳至下一个可用的结果集
arraysize	指定使用 fetchmany()获取的行数，默认为 1
setinputsizes(sizes)	设置在调用 execute*()方法时分配的内存区域大小
setoutputsize(sizes)	设置列缓冲区大小，对大数据列如 LONGS 和 BLOBS 尤其有用

12.2　使用SQLite

与许多其他数据库管理系统不同，SQLite 不是一个客户端/服务器结构的数据库引擎，而是一种嵌入式数据库，它的数据库就是一个文件。SQLite 将整个数据库，包括定义、表、索引以及数据本身，作为一个单独的、可跨平台使用的文件存储在主机中。由于 SQLite 本身是由 C 语言编写的，而且体积很小，所以，经常被集成到各种应用程序中。Python 就内置了 SQLite3，所以，在 Python 中使用 SQLite，不需要安装任何模块，直接使用。

12.2.1　创建数据库文件

由于 Python 中已经内置了 SQLite3，所以可以直接使用 import 语句导入 SQLite3 模块。Python 操作数据库的通用流程如图 12.1 所示。

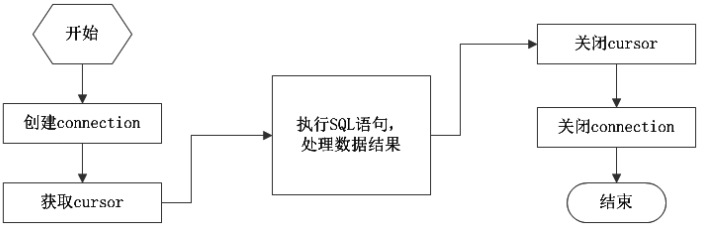


图 12.1　操作数据库流程

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

创建一个 mrsoft.db 的数据库文件，然后执行 SQL 语句创建一个 user 表（用户表），user 表包含 id 和 name 两个字段。具体代码如下：

```
01 import sqlite3
02 # 连接到 SQLite 数据库
03 # 数据库文件是 mrsoft.db，如果文件不存在，会自动在当前目录创建
04 conn = sqlite3.connect('mrsoft.db')
05 # 创建一个 Cursor
06 cursor = conn.cursor()
07 # 执行一条 SQL 语句，创建 user 表
08 cursor.execute('create  table  user (id int(10) primary key, name varchar(20))')
09 # 关闭游标
10 cursor.close()
11 # 关闭 Connection
12 conn.close()
```

在上述代码中，使用 sqlite3.connect()方法连接 SQLite 数据库文件 mrsoft.db，由于 mrsoft.db 文件并不存在，所以会在本实例 Python 代码同级目录下创建 mrsoft.db 文件，该文件包含了 user 表的相关信息。mrsoft.db 文件所在目录如图 12.2 所示。

计算机 ▸ 程序 (D:) ▸ PythonStudy ▸ Code ▸ SL ▸ 12
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
组织 包含到库中 共享 新建文件夹
收藏夹
下载
桌面
最近访问的位
名称 修改日期 类型 大小
01.py 2017/11/20 11:45 Python File 1 KB
mrsoft.db 2017/11/20 11:49 Data Base File 12 KB

图 12.2　mrsoft.db 文件所在目录

> **说明：**再次运行实例时，会提示错误信息：sqlite3.OperationalError:table user alread exists。这是因为 user 表已经存在。

12.2.2　操作SQLite

1．新增用户数据信息

为了向数据表中新增数据，可以使用如下 SQL 语句：

```
insert into 表名(字段名 1,字段名 2,…,字段名 n)  values (字段值 1,字段值 2,…,字段值 n)
```

在 user 表中，有 2 个字段，字段名分别为 id 和 name。而字段值需要根据字段的数据类型来赋值，如 id 是一个长度为 10 的整型数据，name 是长度为 20 的字符串型数据。向 user 表中插入 3 条用户信息记录，则 SQL 语句如下：

```
01 cursor.execute('insert into user (id, name) values ("1", "MRSOFT")')
02 cursor.execute('insert into user (id, name) values ("2", "Andy")')
03 cursor.execute('insert into user (id, name) values ("3", "明日科技小助手")')
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

下面通过一个实例介绍一下向 SQLite 数据库中插入数据的流程。由于在实例中已经创建了 user 表，所以本实例可以直接操作 user 表，向 user 表中插入 3 条用户信息。此外，由于是新增数据，需要使用 commit()方法提交事务。因为对于增加、修改和删除操作，使用 commit()方法提交事务后，如果相应操作失败，可以使用 rollback()方法回滚到操作之前的状态。新增用户数据信息具体代码如下：

```
01 import sqlite3
02 # 连接到 SQLite 数据库
03 # 数据库文件是 mrsoft.db
04 # 如果文件不存在，会自动在当前目录创建
05 conn = sqlite3.connect('mrsoft.db')
06 # 创建一个 Cursor
07 cursor = conn.cursor()
08 # 执行一条 SQL 语句，插入一条记录
09 cursor.execute('insert into user (id, name) values ("1", "MRSOFT")')
10 cursor.execute('insert into user (id, name) values ("2", "Andy")')
11 cursor.execute('insert into user (id, name) values ("3", "明日科技小助手")')
12 # 关闭游标
13 cursor.close()
14 # 提交事务
15 conn.commit()
16 # 关闭 Connection
17 conn.close()
```

向 user 表插入数据

提交事务

运行该实例，会向 user 表中插入 3 条记录。为验证程序是否正常运行，可以再次运行，如果提示如下信息，说明插入成功（因为 user 表中已经保存了上一次插入的记录，所以再次插入会报错）。

```
sqlite3.IntegrityError: UNIQUE constraint failed: user.id
```

2．查看用户数据信息

查找 user 表中的数据可以使用如下 SQL 语句：

```
select  字段名 1,字段名 2,字段名 3,… from 表名  where 查询条件
```

查看用户信息的代码与插入数据信息大致相同，不同点在于使用的 SQL 语句不同。此外，查询数据时通常使用如下 3 种方式。

- fetchone()：获取查询结果集中的下一条记录。
- fetchmany(size)：获取指定数量的记录。
- fetchall()：获取结构集的所有记录。

下面通过一个实例来学习这 3 种查询方式的区别。分别使用 fetchone、fetchmany 和 fetchall 3 种方式查询用户信息，具体代码如下：

```
01 import sqlite3
02 # 连接到 SQLite 数据库,数据库文件是 mrsoft.db
03 conn = sqlite3.connect('mrsoft.db')
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
04 # 创建一个 Cursor
05 cursor = conn.cursor()
06 # 执行查询语句
07 cursor.execute('select * from user')
08 # 获取查询结果
09 result1 = cursor.fetchone()
10 print(result1)
11
12 # 关闭游标
13 cursor.close()
14 # 关闭 Connection
15 conn.close()
```

获取查询结果的语句块

使用 fetchone()方法返回的 result1 为一个元组，运行结果如下：

```
(1,'MRSOFT')
```

（1）修改实例代码，将获取查询结果的语句块代码修改为：

```
01 result2 = cursor.fetchmany(2) # 使用 fetchmany 方法查询多条数据
02 print(result2)
```

使用 fetchmany()方法传递一个参数，其值为 2，默认为 1。返回的 result2 为一个列表，列表中包含 2 个元组，运行结果如下：

```
[(1,'MRSOFT'),(2,'Andy')]
```

（2）修改实例代码，将获取查询结果的语句块代码修改为：

```
01 result3 = cursor.fetchall() # 使用 fetchmany 方法查询多条数据
02 print(result3)
```

使用 fetchall()方法返回的 result3 为一个列表，列表中包含所有 user 表中数据组成的元组，运行结果如下：

```
[(1,'MRSOFT'),(2,'Andy'),(3,'明日科技')]
```

（3）修改实例代码，将获取查询结果的语句块代码修改为：

```
01 cursor.execute('select * from user where id > ?',(1,))
02 result3 = cursor.fetchall()
03 print(result3)
```

在 select 查询语句中，使用问号作为占位符代替具体的数值，然后使用一个元组来替换问号（注意，不要忽略元组中最后的逗号）。上述查询语句等价于：

```
cursor.execute('select * from user where id > 1')
```

运行结果如下：

```
[(2,'Andy'),(3,'明日科技')]
```

说明：使用占位符的方式可以避免 SQL 注入的风险，推荐使用这种方式。

3．修改用户数据信息

修改 user 表中的数据可以使用如下 SQL 语句：

```
update  表名  set 字段名 = 字段值  where 查询条件
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

下面通过一个实例来学习如何修改表中的数据。将 SQLite 数据库中 user 表的 ID 为 1 的数据 name 字段值“mrsoft”修改为“mr”，并使用 fetchall 获取表中的所有数据。具体代码如下：

```
01 import sqlite3
02 # 连接到 SQLite 数据库,数据库文件是 mrsoft.db
03 conn = sqlite3.connect('mrsoft.db')
04 # 创建一个 Cursor
05 cursor = conn.cursor()
06 cursor.execute('update user set name = ? where id = ?',('MR',1))
07 cursor.execute('select * from user')
08 result = cursor.fetchall()
09 print(result)
10 # 关闭游标
11 cursor.close()
12 # 提交事务
13 conn.commit()
14 # 关闭 Connection
15 conn.close()
```

运行结果如下：

```
[(1, 'MR'), (2, 'Andy'), (3, '明日科技小助手')]
```

4．删除用户数据信息

查找 user 表中的数据可以使用如下 SQL 语句：

```
delete  from 表名  where 查询条件
```

下面通过一个实例来学习如何删除表中的数据。将 SQLite 数据库中 user 表的 ID 为 1 的数据删除，并使用 fetchall 获取表中的所有数据，查看删除后的结果。具体代码如下：

```
01 import sqlite3
02 # 连接到 SQLite 数据库,数据库文件是 mrsoft.db
03 conn = sqlite3.connect('mrsoft.db')
04 # 创建一个 Cursor
05 cursor = conn.cursor()
06 cursor.execute('delete from user where id = ?',(1,))
07 cursor.execute('select * from user')
08 result = cursor.fetchall()
09 print(result)
10 # 关闭游标
11 cursor.close()
12 # 提交事务
13 conn.commit()
14 # 关闭 Connection
15 conn.close()
```

执行上述代码后，user 表中 ID 为 1 的数据将被删除。运行结果如下：

```
[(2, 'Andy'), (3, '明日科技小助手')]
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

12.3　使用MySQL

12.3.1　下载和安装MySQL

MySQL 是一款开源的数据库软件，由于其免费特性得到了全世界用户的喜爱，是目前使用人数最多的数据库。下面将详细讲解如何下载和安装 MySQL 数据库。

1．下载 MySQL

（1）在浏览器的地址栏中输入地址“https://dev.mysql.com/downloads/windows/installer/5.7.html”，并按〈Enter〉键，将进入 MySQL 5.7 的下载页面，选择离线安装包，如图 12.3 所示。

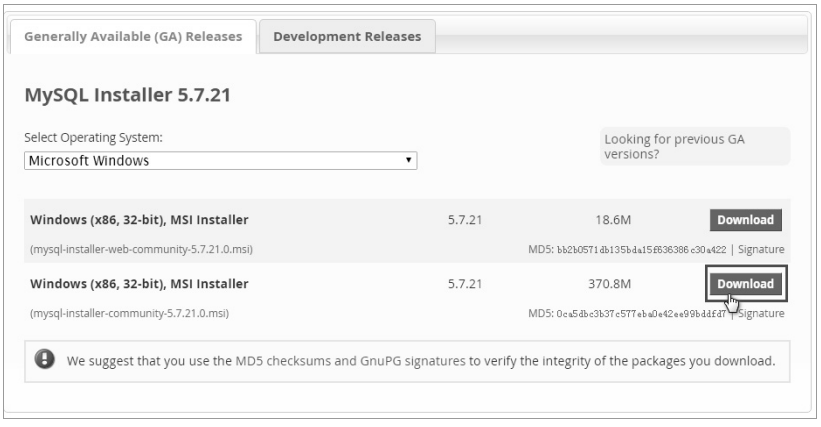


图 12.3　下载 MySQL

（2）单击“Download”按钮，进入下载页面，如果有 MySQL 的账户，可以单击 Login 按钮，登录账户后下载；如果没有账户，可以直接单击下方的“No thanks, just start my download.”超链接，跳过注册步骤，直接下载，如图 12.4 所示。

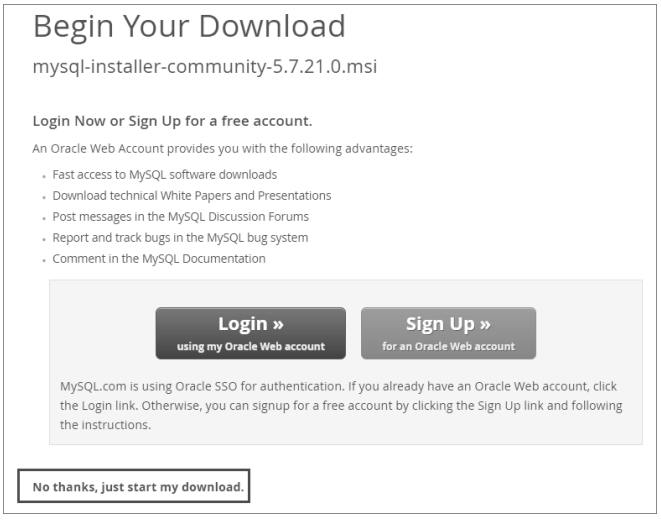


图 12.4　不注册下载

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2．安装 MySQL

下载完成以后，开始安装 MySQL。双击安装文件，在打开的界面中勾选“I accept the license terms”复选框，单击“next”按钮，进入选择设置类型界面。在选择设置中有 5 种类型，说明如下。

- Developer Default：安装 MySQL 服务器以及开发 MySQL 应用所需的工具。工具包括开发和管理服务器的 GUI 工作台、访问操作数据的 Excel 插件、与 Visual Studio 集成开发的插件、通过 NET/Java/C/C++/OBDC 等访问数据的连接器、例子和教程、开发文档。
- Server only：仅安装 MySQL 服务器，适用于部署 MySQL 服务器。
- Client only：仅安装客户端，适用于基于已存在的 MySQL 服务器进行 MySQL 应用开发的情况。
- Full：安装 MySQL 所有可用组件。
- Custom：自定义需要安装的组件。

MySQL 会默认选择“Developer Default”类型，这里我们选择纯净的“Server only”类型，如图 12.5 所示，然后一直选择默认选项安装。

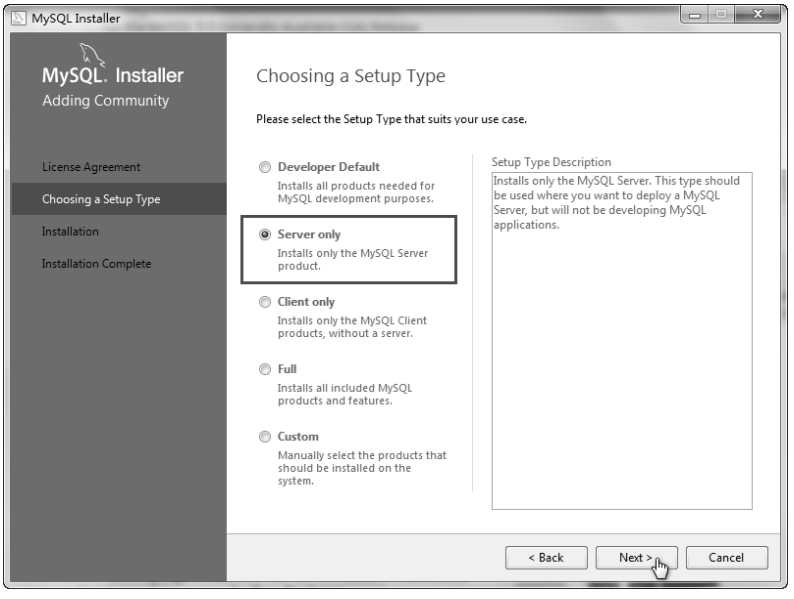


图 12.5　选择安装类型

3．设置环境变量

安装完成以后，默认的安装路径是“C:\Program Files\MySQL\MySQL Server 5.7\bin”。下面设置环境变量，以便在任意目录下使用 MySQL 命令。用鼠标右键单击“计算机”，在弹出的快捷菜单中选择“属性”命令，然后选择“高级系统设置”选项卡，单击“环境变量”按钮，在弹出的对话框中选择“PATH”选项，单击“编辑”按钮，在“变量值”文本框中输入“C:\Program Files\MySQL\MySQL Server 5.7\bin”，如图 12.6 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 12.6　设置环境变量

4．启动 MySQL

使用 MySQL 数据库前，需要先启动 MySQL。在 cmd 窗口中，输入命令“net start mysql57”启动 MySQL 5.7。启动成功后，使用账户和密码进入 MySQL。输入命令“mysql –u root -p”，接着提示“Enter password:”，输入密码“root”即可进入 MySQL，如图 12.7 所示。

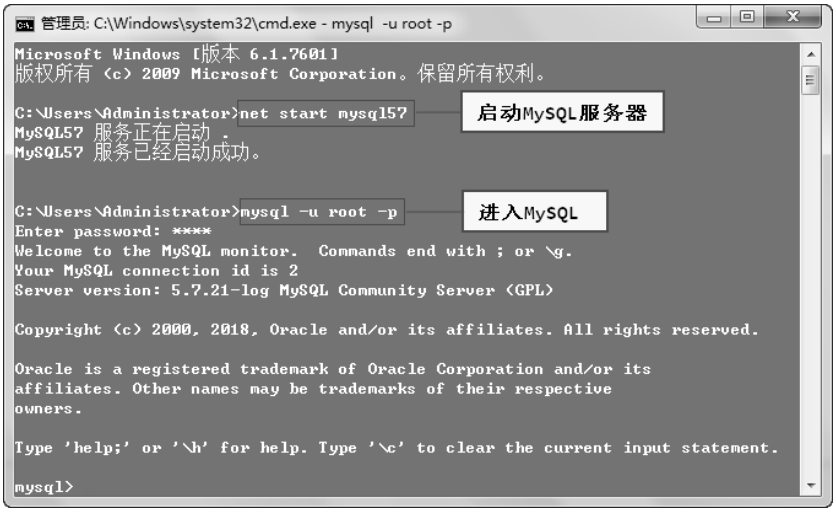

管理员: C:\Windows\system32\cmd.exe - mysql -u root -p
Microsoft Windows [版本 6.1.7601]
版权所有 (c) 2009 Microsoft Corporation。保留所有权利。
C:\Users\Administrator>net start mysql57
启动MySQL服务器
MySQL57 服务正在启动 .
MySQL57 服务已经启动成功。
C:\Users\Administrator>mysql -u root -p
进入MySQL
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.21-log MySQL Community Server (GPL)
Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql>

图 12.7　启动 MySQL

5．使用 Navicat for MySQL 管理软件

在命令提示符下操作 MySQL 数据库的方式对初学者并不友好，而且需要有专业的 SQL 语言知识，所以各种 MySQL 图形化管理工具应运而生，其中 Navicat for MySQL 就是一个广受好评的桌面版 MySQL 数据库管理和开发工具。它使用图形化的用户界面，可以让用户使用和管理更为轻松。官方网址：https://www.navicat.com.cn。

首先下载、安装 Navicat for MySQL，然后新建 MySQL 连接，如图 12.8 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

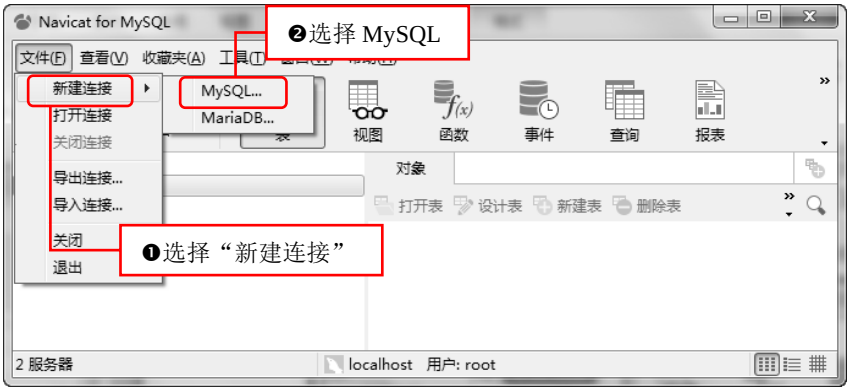


图 12.8　新建 MySQL 连接

接下来输入连接信息。输入“连接名”为“studyPython”，输入“主机名或 IP 地址”为“localhost”或“127.0.0.1”，输入密码为“root”，如图 12.9 所示。

图 12.9　输入连接信息

单击“确定”按钮，创建完成。此时，双击“localhost”，即可进入“localhost”数据库，如图 12.10 所示。

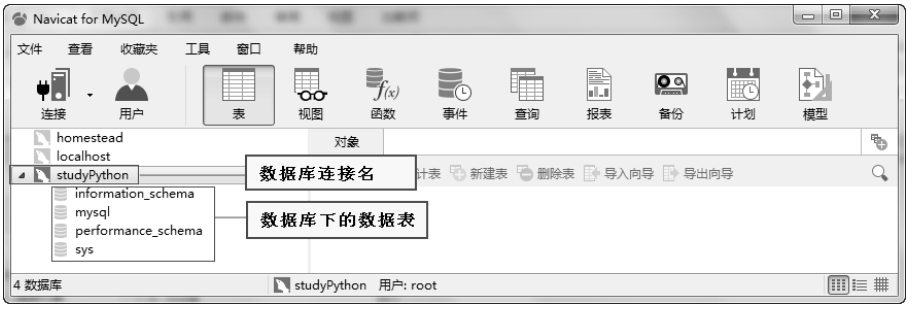


图 12.10　Navicat 主页

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

下面使用 Navicat 创建一个名为“mrsoft”的数据库。用鼠标右键单击“studyPython”，在弹出的快捷菜单中选择“新建数据库”，在弹出的对话框中填写数据库信息，如图 12.11 所示。

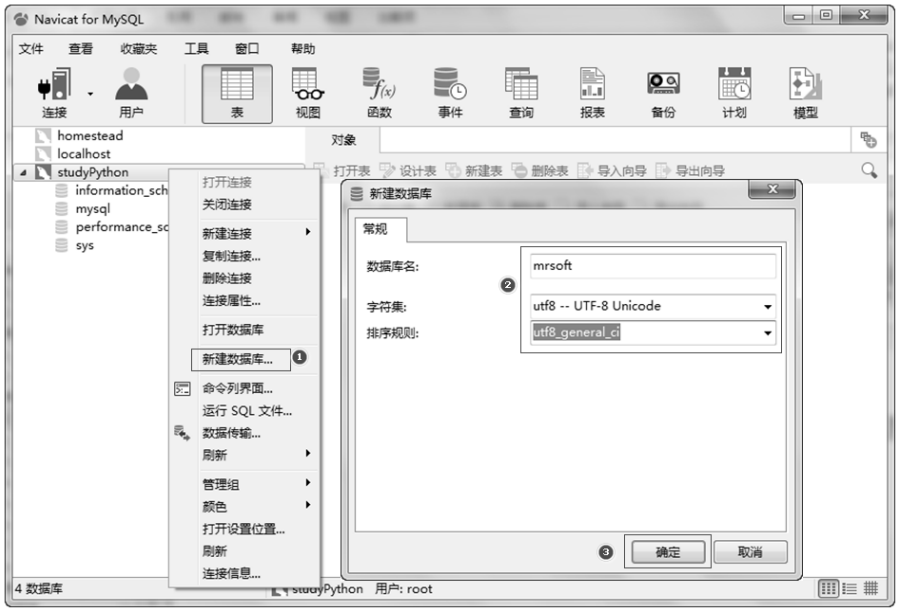


图 12.11　创建数据库

说明：Navicat for MySQL 的更多操作请查阅相关资料。

12.3.2　安装PyMySQL

由于 MySQL 服务器以独立的进程运行，并通过网络对外服务，所以，需要支持 Python 的 MySQL 驱动来连接 MySQL 服务器。在 Python 中支持 MySQL 的数据库模块有很多，我们选择使用 PyMySQL。

PyMySQL 的安装比较简单，在 cmd 中运行如下命令：

```
pip install PyMySQL
```

运行结果如图 12.12 所示。

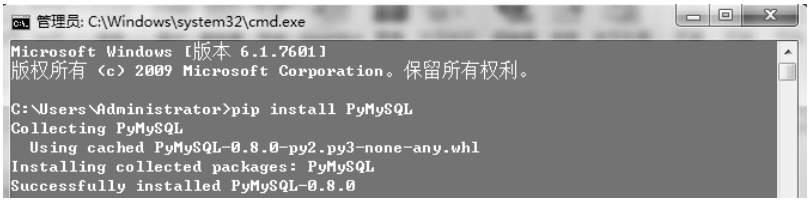


图 12.12　安装 PyMySQL

12.3.3　连接数据库

使用数据库的第一步是连接数据库，接下来使用 PyMySQL 连接数据库。由于 PyMySQL

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

也遵循 Python Database API 2.0 规范，所以操作 MySQL 数据库的方式与 SQLite 相似。我们可以通过类比的方式来学习。

前面我们已经创建了一个 MySQL 连接“studyPython”，并且在安装数据库时设置了数据库的用户名“root”和密码“root”。下面就通过以上信息，使用 connect()方法连接 MySQL 数据库。具体代码如下：

```
01 import pymysql
02
03 # 打开数据库连接,参数 1:主机名或 IP; 参数 2: 用户名; 参数 3: 密码; 参数 4: 数据库名称
04 db = pymysql.connect("localhost", "root", "root", "studyPython")
05 # 使用 cursor()方法创建一个游标对象 cursor
06 cursor = db.cursor()
07 # 使用 execute()方法执行 SQL 查询
08 cursor.execute("SELECT VERSION()")
09 # 使用 fetchone()方法获取单条数据
10 data = cursor.fetchone()
11 print ("Database version : %s " % data)
12 # 关闭数据库连接
13 db.close()
```

在上述代码中，首先使用 connect()方法连接数据库，然后使用 cursor()方法创建游标，接着使用 execute()方法执行 SQL 语句查看 MySQL 数据库版本，然后使用 fetchone()方法获取数据，最后使用 close()方法关闭数据库连接。运行结果如下：

```
Database version : 5.7.21-log
```

12.3.4 创建数据表

数据库连接成功以后，我们就可以为数据库创建数据表了。下面通过一个实例，使用 execute()方法来为数据库创建表 books（图书表）。books 表包含 id（主键）、name（图书名称），category（图书分类）、price（图书价格）和 publish_time（出版时间）5 个字段。创建 books 表的 SQL 语句如下：

```
CREATE TABLE books (
  id int(8) NOT NULL AUTO_INCREMENT,
  name varchar(50) NOT NULL,
  category varchar(50) NOT NULL,
  price decimal(10,2) DEFAULT NULL,
  publish_time date DEFAULT NULL,
  PRIMARY KEY (id)
) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

在创建数据表前，使用如下语句：

```
DROP TABLE IF EXISTS `books`;
```

如果 mrsoft 数据库中已经存在 books，那么先删除 books，然后再创建 books 数据表。具体代码如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
01 import pymysql
02
03 # 打开数据库连接
04 db = pymysql.connect("localhost", "root", "root", "mrsoft")
05 # 使用 cursor()方法创建一个游标对象 cursor
06 cursor = db.cursor()
07 # 使用 execute()方法执行 SQL，如果表存在则删除
08 cursor.execute("DROP TABLE IF EXISTS books")
09 # 使用预处理语句创建表
10 sql = """
11 CREATE TABLE books (
12   id int(8) NOT NULL AUTO_INCREMENT,
13   name varchar(50) NOT NULL,
14   category varchar(50) NOT NULL,
15   price decimal(10,2) DEFAULT NULL,
16   publish_time date DEFAULT NULL,
17   PRIMARY KEY (id)
18 ) ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
19 """
20 # 执行 SQL 语句
21 cursor.execute(sql)
22 # 关闭数据库连接
23 db.close()
```

运行上述代码后，mrsoft 数据库下就已经创建了一个 books 表。打开 Navicat（如果已经打开按下<F5>键刷新），发现 mrsoft 数据库下多了一个 books 表，用鼠标右键单击“books”，在弹出的快捷菜单中选择“设计表”，效果如图 12.13 所示。

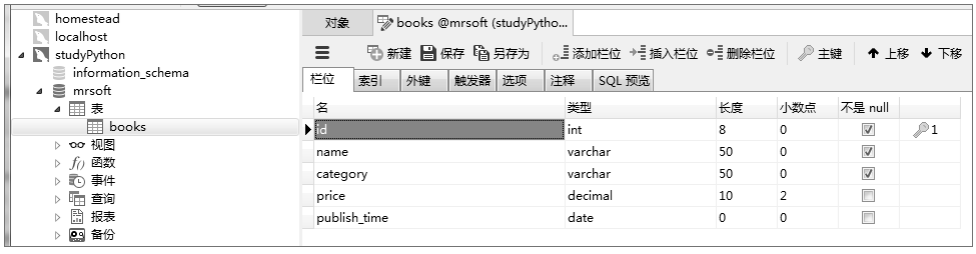


图 12.13　创建 books 表效果

12.3.5　操作MySQL数据表

MySQL 数据表的操作主要包括数据的新增、删除、修改和查找，与操作 SQLite 类似，这里我们通过一个实例讲解如何向 books 表中新增数据，至于修改、查找和删除数据则不再赘述。在向books表中插入图书数据时，可以使用execute()方法添加一条记录，也可以使用executemany()方法批量添加多条记录。executemany()方法的格式如下。

```
executemany(operation, seq_of_params)
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- operation：操作的 SQL 语句。
- seq_of_params：参数序列。

executemany()方法批量添加多条记录的具体代码如下：

```
01 import pymysql
02
03 # 打开数据库连接
04 db = pymysql.connect("localhost", "root", "root", "mrsoft",charset="utf8")
05 # 使用 cursor()方法获取操作游标
06 cursor = db.cursor()
07 # 数据列表
08 data = [("零基础学 Python",'Python','79.80','2018-5-20'),
09        ("Python 从入门到精通",'Python','69.80','2018-6-18'),
10        ("零基础学 PHP",'PHP','69.80','2017-5-21'),
11        ("PHP 项目开发实战入门",'PHP','79.80','2016-5-21'),
12        ("零基础学 Java",'Java','69.80','2017-5-21'),
13        ]
14 try:
15     # 执行 SQL 语句，插入多条数据
16     cursor.executemany("insert into books(name, category, price, publish_time)
values (%s,%s,%s,%s)", data)
17     # 提交数据
18     db.commit()
19 except:
20     # 发生错误时回滚
21     db.rollback()
22
23 # 关闭数据库连接
24 db.close()
```

在上述代码中，特别注意以下几点：

- 使用 connect()方法连接数据库时，额外设置字符集 charset=utf-8，可以防止插入中文时出错。
- 在使用 insert 语句插入数据时，使用%s 作为占位符，可以防止 SQL 注入。

运行上述代码，在 Navicat 中查看 books 表数据，效果如图 12.14 所示。

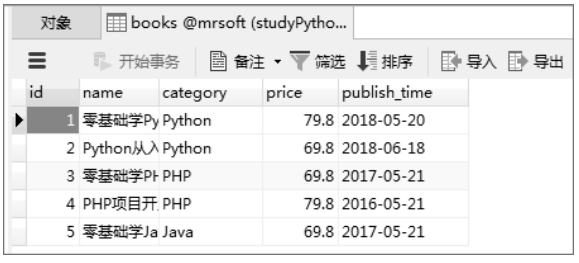


图 12.14　books 表数据

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：（010）88254396；（010）88258888

传　　真：（010）88254397

E－mail：　dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱　电子工业出版社总编办公室

邮　　编：100036

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

Python
编程入门指南 下册

明日科技
编著

视频教学+VIP代码库

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

電子工業出版社

内 容 简 介

本书以初学者为对象，全面介绍了使用 Python 进行程序开发的各种技术。在内容排列上由浅入深，让读者循序渐进掌握编程技术；在内容讲解上结合丰富的图解和形象的比喻，帮助读者理解“晦涩难懂”的技术。全书共分 22 章，包括搭建 Python 开发环境、Python 入门必学、变量和数据类型、流程控制语句、序列的应用、字符串及正则表达式、异常处理及程序调试、函数、面向对象程序设计、模块、文件及目录操作、使用 Python 操作数据库、常用的 GUI 框架、Django Web 框架的使用、Pygame 游戏框架、网络爬虫框架、Flask Web 框架的使用、中华诗词大会（网络版）、图像识别工具、微信跳一跳游戏的辅助工具、汽车之家图片抓取工具和巡店宝（企业商品运营利器）。全书内容丰富，项目贴近实际开发，适合读者自学。

本书适用于 Python 的爱好者、初学者和中级开发人员，也可作为大、中专院校和培训机构的教材。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

Python 编程入门指南：全 2 册 / 明日科技编著. —北京：电子工业出版社，2019.5

ISBN 978-7-121-35797-8

Ⅰ. ①P… Ⅱ. ①明… Ⅲ. ①软件工具－程序设计 Ⅳ. ①TP311.561

中国版本图书馆 CIP 数据核字（2018）第 275714 号

策划编辑：徐　岩
责任编辑：张　毅　　特约编辑：田学清
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×1092　1/16　　印张：26.5　　字数：679 千字
版　　次：2019 年 5 月第 1 版
印　　次：2019 年 5 月第 1 次印刷
定　　价：80.00 元（全 2 册）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：（010）57565805。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

前 言

Python 是一门有条理的和强大的面向对象的程序设计语言，因其具有丰富和强大的库，还可以把用其他编程语言制作的各种模块（尤其是 C/C++）很轻松地结合在一起，进行高效开发，Python 语言已经成为最受欢迎的程序设计语言之一。Python 语言简单、易学、易用，适合在 Web、数据分析、游戏、科学计算等领域进行编程开发。本书是一本面向 Python 零基础读者的入门书籍，按照基础入门→进阶提升→框架应用→项目开发的循序渐进的过程进行学习和实践。

本书特点

- 结构合理，适合自学。本书定位以初学者为主，在内容安排上充分体现了自学的特点，内容循序渐进，由浅入深，覆盖了编程的各方面技术。
- 视频讲解，全面到位。视频讲解是本书的特色之一，本书提供了近 23 小时的视频同步课程。每个知识点都力求讲透、讲全，在对各知识点的讲解过程中提供了很多编程经验和技巧。
- 比喻生动，容易理解。在编程领域中有许多“晦涩难懂”的名词或技术难以理解，本书为了帮助读者理解，采用了比喻的手法，并结合示意图，帮助读者掌握这些技术。
- 实例丰富，一学就会。书中在介绍知识点的时候，都伴有大量的实例或示例，并且提供实例的设计过程。读者可以按照步骤做出实例，巩固所学的知识。
- 精彩栏目，贴心提醒。本书在各章节中穿插了大量的注意、说明、技巧等栏目，让读者可以在学习过程中更轻松地理解相关知识点及概念，并轻松地掌握个别技术的应用技巧。

读者对象

- 初学编程的自学者。
- 大、中专院校的老师和学生。
- 初、中级程序开发人员。
- 想学习编程的企事业单位在职人员。
- 编程爱好者。
- 相关培训机构的老师和学员。
- 程序测试及维护人员。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

如何研读本书

为了快速掌握本书内容，读者可以按照以下方法研读本书。

- 第一，按顺序阅读。书中的内容是按照由浅入深的顺序精心组织的，对于初学者来说，强烈建议你不要急于学习后面复杂的章节，这样容易顾此失彼，影响学习的自信心。
- 第二，学练结合。对于程序开发人员来说，“理解”和“会做”是两个不同的层次。只有“会做”，也就是按照自己的思维、想法将程序开发出来才能说明完全掌握了知识。
- 第三，多思考。在掌握了一项技术后，要想想该技术还能够解决哪些问题，这样可以使我们掌握的有限的技术发挥最大的功效，在工作中能够充分发挥自己的才能。
- 第四，充满信心。诚然，学习编程是一个艰苦的过程，即使再优秀的图书，也不可能提供所有的经验，解决编程中的所有问题。因为每个人学习编程的经历不同，所积累的经验和遇到的问题不可能完全相同，这就需要我们充满信心，以“不放弃”的态度克服遇到的问题，相信问题终究会解决。

读者服务

为了方便解决本书中的疑难问题，我们提供了多种服务方式，并由作者团队提供在线技术指导和社区服务，服务方式如下。

- 服务网站：www.mingrisoft.com。
- 服务邮箱：mingrisoft@mingrisoft.com。
- 企业 QQ：4006751066。
- QQ 群：539340057。
- 服务电话：400-675-1066、0431-84978981。

致读者

本书由 Python 程序开发团队策划并组织编写，主要编写人员有赛奎春、白兆松、王欢、王小科、冯春龙、王国辉、李磊、杨柳、岳彩龙、潘建羽、葛忠月、李春林、张宝华、杨丽、梁英等。在编写本书的过程中，我们始终坚持“坚韧、创新、博学、笃行”的企业理念，以科学、严谨的态度，力求精益求精，但错误、疏漏之处在所难免，敬请广大读者批评指正。

编　者

2018 年 10 月

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

目 录

第 4 篇　框架开发篇

第 13 章　常用的 GUI 框架 191

13.1　初识 GUI 191

13.1.1　什么是 GUI 191

13.1.2　常用的 GUI 框架 192

13.2　wxPython 框架的使用 192

13.2.1　安装 wxPython 192

13.2.2　创建一个 wx.App 的子类 193

13.2.3　直接使用 wx.App 193

13.2.4　使用 wx.Frame 框架 194

13.2.5　常用控件 195

13.2.6　BoxSizer 布局 200

13.2.7　事件处理 204

13.3　PyQt 框架的使用 206

13.3.1　安装 PyQt 206

13.3.2　使用第三方开发工具 207

13.3.3　创建主窗体 210

13.3.4　常用控件 213

13.3.5　布局管理 217

13.3.6　信号与槽的关联 218

13.3.7　资源文件的使用 220

13.4　小结 225

第 14 章　Django Web 框架的使用 226

14.1　Django 框架的使用 226

14.1.1　安装 Django Web 框架 226

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

14.1.2 创建一个 Django 项目 228
14.1.3 创建 App 230
14.1.4 数据模型(models) 230
14.1.5 管理后台 235
14.1.6 路由(urls) 236
14.1.7 表单（forms） 238
14.1.8 视图(views) 239
14.1.9 Django 模板 242
14.2 小结 243
第 15 章 Pygame 游戏框架 244
15.1 初识 Pygame 244
15.1.1 安装 Pygame 244
15.1.2 Pygame 常用模块 245
15.2 Pygame 基本使用 247
15.3 小结 252
第 16 章 网络爬虫框架 253
16.1 初识网络爬虫 253
16.1.1 网络爬虫概述 253
16.1.2 网络爬虫的分类 253
16.1.3 网络爬虫的基本原理 254
16.2 网络爬虫的常用技术 255
16.2.1 Python 的网络请求 255
16.2.2 请求头部处理 258
16.2.3 网络超时 259
16.2.4 代理服务 260
16.2.5 HTML 解析之 BeautifulSoup 261
16.3 网络爬虫开发常用框架 264
16.3.1 Scrapy 爬虫框架 264
16.3.2 Crawley 爬虫框架 264
16.3.3 PySpider 爬虫框架 265
16.4 Scrapy 爬虫框架的使用 265
16.4.1 搭建 Scrapy 爬虫框架 265
16.4.2 创建 Scrapy 项目 268

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

16.4.3 创建爬虫 268
16.4.4 获取数据 270
第 17 章 Flask Web 框架的使用 273
17.1 Web 框架简介 273
17.1.1 什么是 Web 框架 273
17.1.2 常用的 Web 框架 274
17.2 Flask 框架的使用 274
17.2.1 安装虚拟环境 275
17.2.2 安装 Flask 框架 276
17.2.3 第一个 Flask 程序 277
17.2.4 开启调试模式 278
17.2.5 路由 278
17.2.6 静态文件 280
17.2.7 模板 281
17.3 小结 284

第 5 篇 Web 开发篇

第 18 章 中华诗词大会（网络版）（Django 框架+Bootstrap+MySQL+Redis 实现） 285
18.1 项目概述 285
18.2 系统功能设计 285
18.2.1 了解项目整体开发流程 285
18.2.2 系统功能结构 286
18.2.3 系统业务流程 287
18.2.4 系统预览 287
18.3 系统开发必备 288
18.3.1 系统开发环境 288
18.3.2 项目目录结构 289
18.4 数据库设计 290
18.4.1 数据库概要说明 290
18.4.2 数据表模型 291
18.5 用户登录模块设计 293
18.5.1 用户登录模块概述 293
18.5.2 使用 Django 默认的用户授权机制实现普通登录 294

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

18.5.3 基于 Oauth 2.0 的微信登录的实现 302
18.5.4 机构注册功能的实现 304
18.6 核心答题功能的设计 310
18.6.1 答题首页设计 310
18.6.2 比赛详情页面 313
18.6.3 答题功能的实现 316
18.6.4 提交答案 321
18.6.5 批量录入题库 324

第 6 篇　商业项目篇

第 19 章　图像识别工具 331
19.1 项目概述 331
19.2 设计流程 331
19.3 系统预览 332
19.4 开发工具准备 333
19.5 开发前的准备工作 334
19.5.1 申请百度 AI 接口 334
19.5.2 urllib、urllib.request 模块 335
19.5.3 JSON 模块 336
19.6 图像识别工具的开发 337
19.6.1 根据原型设计制作窗体 337
19.6.2 添加分类 337
19.6.3 选择要识别的图片 338
19.6.4 图像识别 340
19.6.5 复制识别结果到剪贴板 341
第 20 章　微信跳一跳游戏的辅助工具 342
20.1 项目概述 342
20.1.1 设计流程 343
20.1.2 系统预览 343
20.1.3 开发工具准备 344
20.2 开发前的准备工作 344
20.2.1 PIL 模块的安装 345
20.2.2 PIL 模块的使用 345

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

20.2.3 matplotlib 模块的使用 346
20.2.4 ADB 工具 347
20.3 微信跳一跳游戏的辅助工具的实现 348
20.3.1 窗体的实现 348
20.3.2 同步手机屏幕的画面 349
20.3.3 选择起点与终点实现跳跃 351
20.3.4 实现重选功能 353
20.4 微信跳一跳游戏辅助工具（自动）的实现 354
20.4.1 同步手机屏幕的画面 354
20.4.2 扫描棋子与棋盘 355
20.4.3 实现自动跳跃 357
第 21 章 汽车之家图片抓取工具 360
21.1 项目概述 360
21.2 设计流程 360
21.2.1 系统预览 361
21.2.2 开发工具准备 363
21.3 汽车之家图片抓取工具的开发 363
21.3.1 主窗体设计 363
21.3.2 创建爬虫类 363
21.3.3 开启爬虫类 365
21.3.4 查看大图功能 368
第 22 章 巡店宝（企业商品运营利器） 369
22.1 项目概述 369
22.2 设计流程 370
22.3 系统预览 370
22.4 开发工具准备 371
22.5 开发前的准备工作 372
22.5.1 selenium 模块的安装 372
22.5.2 下载浏览器驱动 372
22.5.3 selenium 模块的使用 373
22.5.4 selenium 模块的常用方法 373
22.6 巡店宝（京东版）查询工具 374
22.6.1 窗体的实现 374

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

22.6.2 获取京东商品 name_id 377
22.6.3 获取京东商品 json_id 379
22.6.4 浏览器自动加载指定商品的网页 382
22.6.5 分析与显示网页中的图书信息 385
22.6.6 显示目录窗口 390
22.6.7 显示好评、中评及差评信息 393
22.6.8 显示好评度分析窗口 397

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 4 篇　框架开发篇

第 13 章

常用的 GUI 框架

到目前为止，我们所有的输入和输出都只是 IDLE 中的简单文本，不过现代计算机和程序会使用大量的图形，如果我们的程序中也有一些图形就太好了。在这一章中，我们会开始建立一些简单的 GUI。这说明从现在开始，我们的程序看上去就会像你平常熟悉的那些程序一样，将会有窗口、按钮之类的图形。

13.1　初识GUI

13.1.1　什么是 GUI

GUI 是 Graphical User Interface（图形用户界面）的缩写。在 GUI 中，并不只是键入文本和返回文本，用户可以看到窗口、按钮、文本框等图形，而且可以用鼠标单击，还可以通过键盘输入。GUI 是一种与程序交互不同的方式。GUI 的程序有 3 个基本要素：输入、处理和输出，如图 13.1 所示，但它们的输入和输出更丰富、更有趣一些。

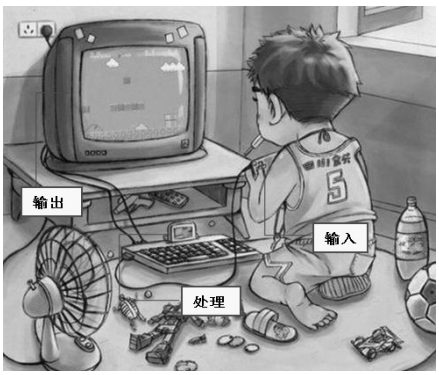


图 13.1　GUI 的 3 要素

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

13.1.2 常用的 GUI 框架

对于 Python 的 GUI 开发，有很多工具包供我们选择。其中一些流行的工具包如表 13.1 所示。

表 13.1 流行的 GUI 工具包

工 具 包	描 述
wxPython	wxPython 是 Python 语言的一套优秀的 GUI 图形库，允许 Python 程序员很方便地创建完整的、功能键全的 GUI 用户界面
Kivy	Kivy 是一个开源工具包，能够让使用相同源代码创建的程序跨平台运行。它主要关注创新型用户界面的开发，如多点触摸应用程序
Flexx	Flexx 是一个纯 Python 工具包，用来创建图形化界面应用程序。其使用 Web 技术进行界面的渲染
PyQt	PyQt 是 Qt 库的 Python 版本，支持跨平台
Tkinter	Tkinter（也叫 Tk 接口）是 Tk 图形用户界面工具包标准的 Python 接口。Tk 是一个轻量级的跨平台图形用户界面（GUI）开发工具
Pywin32	Windows Pywin32 允许你以像 VC 一样的形式来使用 Python 开发 Win32 应用
PyGTK	PyGTK 让你用 Python 轻松创建具有图形用户界面的程序
pyui4win	pyui4win 是一个开源的采用自绘技术的界面库

每个工具包都有其优缺点，所以工具包的选择取决于你的应用场景。本章将详细介绍 wxPython 的使用方法。

13.2 wxPython框架的使用

13.2.1 安装 wxPython

wxPython 是一个成熟且特性丰富的跨平台 GUI 工具包，由 Robin Dunn 及 Harri Pasanen 开发，官方网站：http://wxpython.org。wxPython 的安装非常简单，使用 pip 工具安装 wxPython 只需要一行命令：

```
pip install -U wxPython
```

在 Windows 的 cmd 命令下，使用 pip 安装 wxPython，如图 13.2 所示。

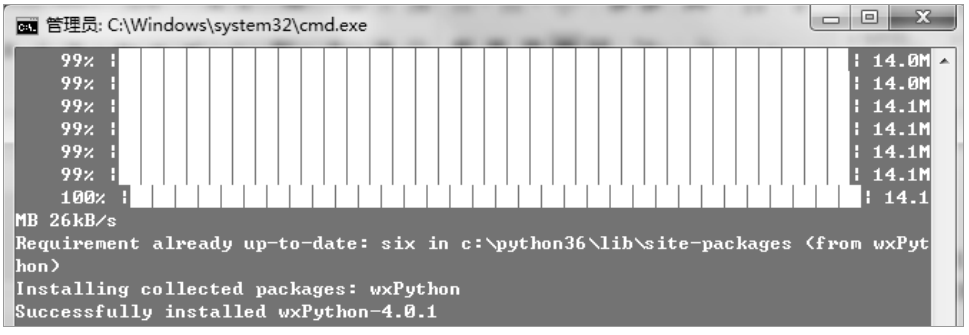


图 13.2 安装 wxPython

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

13.2.2　创建一个 wx.App 的子类

在开始创建应用程序之前，先来创建一个没有任何功能的子类。创建和使用一个 wx.App 子类，需要执行如下 4 个步骤：

（1）定义这个子类。

（2）在定义的子类中写一个 OnInit()初识化方法。

（3）在程序的主要部分创建这个类的一个实例。

（4）调用应用程序实例的 MainLoop()方法，这个方法将程序的控制权转交给 wxPython。

创建一个没有任何功能的子类，具体代码如下：

```
01 # -*- coding:utf-8 -*-
02 import wx           # 导入 wxPython
03 class App(wx.App):
04     # 初始化方法
05     def OnInit(self):
06         frame = wx.Frame(parent=None, title='Hello wyPython') # 创建窗口
07         frame.Show()     # 显示窗口
08         return True      # 返回值
09
10 if __name__ == '__main__':
11     app = App()      # 创建 App 类的实例
12     app.MainLoop()  # 调用 App 类的 MainLoop()主循环方法
```

在上述代码中，定义了一个子类 App()，它继承父类 wx.App，子类中包含一个初始化方法 OnInit()。在主程序中创建类的实例，然后调用 MainLoop()主循环方法。运行结果如图 13.3 所示。

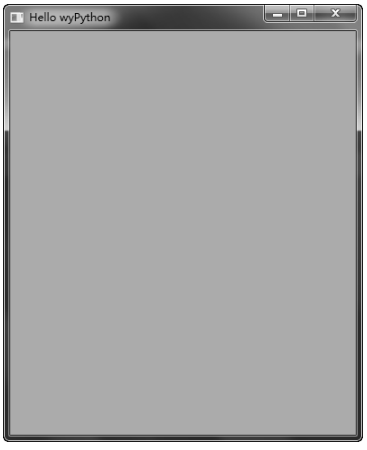


图 13.3　创建子类

13.2.3　直接使用 wx.App

通常，如果在系统中只有一个窗口的话，可以不创建 wx.App 子类，直接使用 wx.App，

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

这个类提供了一个最基本的 OnInit()初始化方法，具体代码如下：

```
01 # -*- coding:utf-8 -*-
02 import wx               # 导入 wxPython
03 app   = wx.App()        # 初始化 wx.App 类
04 frame = wx.Frame(None,title='Hello wyPython')  # 定义了一个顶级窗口
05 frame.Show()            # 显示窗口
06 app.MainLoop()          # 调用 wx.App 类的 MainLoop()主循环方法
```

在上述代码中，wx.App()初始化 wx.App 类，包含了 OnInit()方法。

13.2.4 使用 wx.Frame 框架

在 GUI 中框架通常也称为窗口。框架是一个容器，用户可以将它在屏幕上任意移动，并可对它进行缩放，它通常包含标题栏、菜单等。在 wxPython 中，wx.Frame 是所有框架的父类。当你创建 wx.Frame 的子类时，子类应该调用其父类的构造器 wx.Frame.__init__()。wx.Frame 的构造器语法格式如下：

```
wx.Frame(parent, id=-1, title="", pos=wx.DefaultPosition, size=wx.DefaultSize,
        style=wx.DEFAULT_FRAME_STYLE, name="frame")
```

参数的说明如下。

- parent：框架的父窗口。如果是顶级窗口，这个值是 None。
- id：关于新窗口的 wxPython ID 号。通常设为-1，让 wxPython 自动生成一个新的 ID。
- title：窗口的标题。
- pos：一个 wx.Point 对象，它指定这个新窗口的左上角在屏幕中的位置。在图形用户界面程序中，通常(0,0)是显示器的左上角。这个默认的(-1,-1)将让系统决定窗口的位置。
- size：一个 wx.Size 对象，它指定这个窗口的初始尺寸。这个默认的(-1,-1)将让系统决定窗口的初始尺寸。
- style：指定窗口的类型的常量。可以使用或运算来组合它们。
- name：框架的内在的名字。可以使用它来寻找这个窗口。

创建 wx.Frame 子类的代码如下：

```
01 # -*- coding:utf-8 -*-
02 import wx   # 导入 wxPython
03 class MyFrame(wx.Frame):
04     def __init__(self,parent,id):
05         wx.Frame.__init__(self,parent,id, title="创建 Frame",pos=(100, 100),
size=(300, 300))
06
07 if __name__ == '__main__':
08     app = wx.App()                          # 初始化应用
09     frame = MyFrame(parent=None,id=-1)  # 实例化 MyFrame 类，并传递参数
10     frame.Show()                            # 显示窗口
11     app.MainLoop()                          # 调用 MainLoop()主循环方法
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

在上述代码中，在主程序中调用 MyFrame 类，并且传递 2 个参数。在 MyFrame 类中，自动执行__init__()初始化方法，接收参数。然后调用父类 wx.Frame 的__init__()初始化方法，设置顶级窗口的相关属性。运行结果如图 13.4 所示。

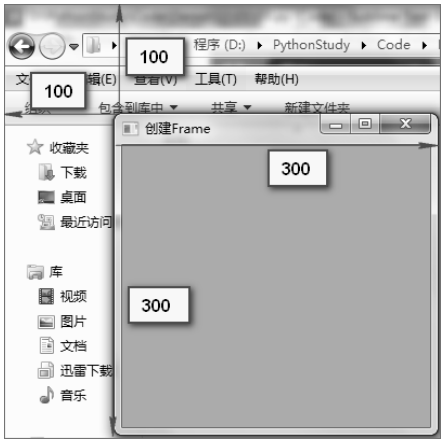


图 13.4　使用 wx.Frame 框架

13.2.5　常用控件

创建完窗口以后，我们可以在窗口内添加一些控件，所谓的控件，就是经常使用的按钮、文本、输入框、单选按钮等。

1．StaticText 文本类

对于所有的 UI 工具来说，最基本的任务就是在屏幕上绘制纯文本。在 wxPython 中，可以使用 wx.StaticText 类来完成。使用 wx.StaticText 能够改变文本的对齐方式、字体和颜色等。wx.StaticText 类的构造函数语法格式如下：

```
 wx.StaticText(parent, id, label, pos=wx.DefaultPosition,size=wx.DefaultSize,
style=0, name="staticText")
```

wx.StaticText 构造函数的参数如下所示。

- parent：父窗口部件。
- id：标识符。使用-1 可以自动创建一个唯一的标识。
- label：显示在静态控件中的文本内容。
- pos：一个 wx.Point 或一个 Python 元组，它是窗口部件的位置。
- size：一个 wx.Size 或一个 Python 元组，它是窗口部件的尺寸。
- style：样式标记。
- name：对象的名字。

实例 01　使用 wx.StaticText 输出 Python 之禅

在 Python 控制台中输入 import this 后，会输出如图 13.5 所示的结果，结果中的英文语句就是通常所说的 Python 之禅。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

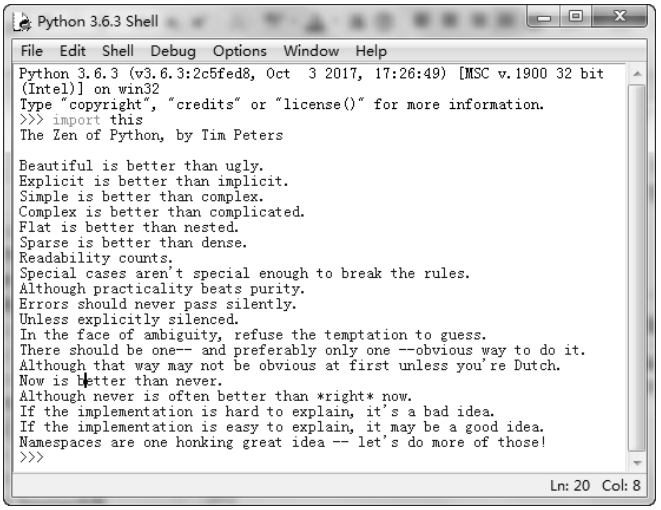

Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import this
The Zen of Python, by Tim Peters
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
Ln: 20 Col: 8

图 13.5　Python 之禅

下面使用 StaticText 类输出中文版的 Python 之禅，具体代码如下：

```
01 # -*- coding:utf-8 -*-
02 import wx
03 class MyFrame(wx.Frame):
04     def __init__(self,parent,id):
05         wx.Frame.__init__(self, parent, id, title = "创建 StaticText 类",
06                           pos=(100, 100), size=(600, 400))
07         panel = wx.Panel(self) # 创建画板
08         # 创建标题，并设置字体
09         title = wx.StaticText(panel, label='Python 之禅——Tim Peters',pos=
(100,20))
10         font  = wx.Font(16, wx.DEFAULT, wx.FONTSTYLE_NORMAL, wx.NORMAL)
11         title.SetFont(font)
12         # 创建文本
13         wx.StaticText(panel, label='优美胜于丑陋',pos=(50,50))
14         wx.StaticText(panel, label='明了胜于晦涩',pos=(50,70))
15         wx.StaticText(panel, label='简洁胜于复杂',pos=(50,90))
16         wx.StaticText(panel, label='复杂胜于凌乱',pos=(50,110))
17         wx.StaticText(panel, label='扁平胜于嵌套',pos=(50,130))
18         wx.StaticText(panel, label='间隔胜于紧凑',pos=(50,150))
19         wx.StaticText(panel, label='可读性很重要',pos=(50,170))
20         wx.StaticText(panel, label='即便假借特例的实用性之名，也不可违背这些规则',
pos=(50,190))
21         wx.StaticText(panel, label='不要包容所有错误，除非你确定需要这样做',
pos=(50,210))
22         wx.StaticText(panel, label='当存在多种可能，不要尝试去猜测',pos=(50,230))
23         wx.StaticText(panel, label='而是尽量找一种，最好是唯一一种明显的解决方案',
pos=(50,250))
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
24        wx.StaticText(panel, label='虽然这并不容易，因为你不是 Python 之父',
pos=(50,270))
25        wx.StaticText(panel, label='做也许好过不做，但不假思索就动手还不如不做',
pos=(50,290))
26        wx.StaticText(panel,label='如果你无法向人描述你的方案,那肯定不是一个好方案;
反之亦然',pos=(50,310))
27        wx.StaticText(panel, label='命名空间是一种绝妙的理念，我们应当多加利用',
pos=(50,330))
28
29 if __name__ == '__main__':
30     app = wx.App()                                  # 初始化应用
31     frame = MyFrame(parent=None,id=-1)              # 实例化 MyFrame 类，并传递参数
32     frame.Show()                                    # 显示窗口
33     app.MainLoop()                                  # 调用主循环方法
```

在上述代码中，使用 panel=wx.Panel(self)来创建画板，并将 panel 作为父类，然后将组件放入窗体中。此外，使用 wx.Font 类来设置字体。创建一个字体实例，需要使用如下的构造函数：

```
wx.Font(pointSize, family, style, weight, underline=False, faceName="",
encoding=wx.FONTENCODING_DEFAULT)
```

参数说明如下。

- pointSize：字体的整数尺寸，单位为磅。
- family：用于快速指定一个字体而无须知道该字体的实际的名字。
- style：指明字体是否倾斜。
- weight：指明字体的醒目程度。
- underline：仅在 Windows 系统下有效，如果取值为 True，则加下画线，False 为无下画线。
- faceName：指定字体名。
- encoding：允许在几个编码中选择一个，大多数情况可以使用默认编码。

运行结果如图 13.6 所示。

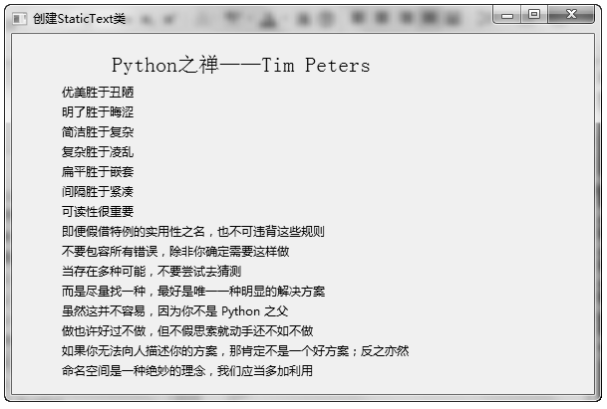


图 13.6　输出 Python 之禅

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2．TextCtrl 输入文本类

wx.StaticText 类只能够用于显示纯粹的静态文本，但是有时需要输入文本与用户进行交互，此时，就需要使用 wx.TextCtrl 类，它允许输入单行和多行文本。它也可以作为密码输入控件，掩饰所按下的按键。

wx.TextCtrl 类的构造函数语法格式如下：

```
wx.TextCtrl(parent, id, value = "", pos=wx.DefaultPosition, size=wx.DefaultSize,
style=0, validator=wx.DefaultValidator name=wx.TextCtrlNameStr)
```

参数 parent、id、pos、size、style 和 name 与 wx.StaticText 构造函数相同，重点看一下其他参数。

- style：单行 wx.TextCtrl 的样式、取值及说明如下。
  - wx.TE_CENTER：控件中的文本居中。
  - wx.TE_LEFT：控件中的文本左对齐。
  - wx.TE_NOHIDESEL：文本始终高亮显示，只适用于 Windows。
  - wx.TE_PASSWORD：不显示所键入的文本，以星号（*）代替显示。
  - wx.TE_PROCESS_ENTER：如果使用该参数，那么当用户在控件内按下回车键时，一个文本输入事件将被触发。否则，按键事件内在的由该文本控件或该对话框管理。
  - wx.TE_PROCESS_TAB：如果指定了这个样式，那么通常的字符事件在<Tab>键按下时创建（一般意味着一个制表符将被插入文本）。否则，<Tab>键由对话框来管理，通常是控件间的切换。
  - wx.TE_READONLY：文本控件为只读，用户不能修改其中的文本。
  - wx.TE_RIGHT：控件中的文本右对齐。
- value：显示在该控件中的初始文本。
- validator：常用于过滤数据以确保只能键入要接收的数据。

实例 02　使用 wx.TextCtrl 实现登录界面

使用 wx.TextCtrl 类和 wx.StaticText 类实现一个包含用户名和密码的登录界面。具体代码如下：

```
01 # -*- coding:utf-8 -*-
02 import wx
03 class MyFrame(wx.Frame):
04     def __init__(self,parent,id):
05         wx.Frame.__init__(self, parent,id, title="创建 TextCtrl 类",size=(400,
300))
06         #创建面板
07         panel = wx.Panel(self)
08         # 创建文本和输入框
09         self.title = wx.StaticText(panel ,label="请输入用户名和密码",pos=(140,20))
10         self.label_user = wx.StaticText(panel,label="用户名:",pos=(50,50) )
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
11        self.text_user = wx.TextCtrl(panel,pos=(100,50),size=(235,25),style=
wx.TE_LEFT)
12        self.label_pwd = wx.StaticText(panel,pos=(50,90),label="密   码:")
13        self.text_password  =  wx.TextCtrl(panel,pos=(100,90),size=(235,25),
style=wx.TE_PASSWORD)
14
15 if __name__ == '__main__':
16     app = wx.App()                              # 初始化应用
17     frame = MyFrame(parent=None,id=-1)          # 实例化 MyFrame 类，并传递参数
18     frame.Show()                                # 显示窗口
19     app.MainLoop()                              # 调用主循环方法
```

在上述代码中，使用 wx.TextCtrl 类生成用户名，并且设置控件中的文本左对齐。使用 wx.TextCtrl 类生成密码，并且设置文本用星号代替。运行结果如图 13.7 所示。

图 13.7　生成“用户名”和“密码”文本框

3．Button 按钮类

按钮是 GUI 界面中应用最为广泛的控件，它常用于捕获用户生成的单击事件，其最明显的用途是触发绑定到一个处理函数。

wxPython 类库提供不同类型的按钮。其中最简单、常用的是 wx.Button 类。wx.Button 类的构造函数语法格式如下：

```
wx.Button(parent, id, label, pos, size=wxDefaultSize, style=0, validator,
name="button")
```

wx.Button 的参数与 wx.TextCtrl 的参数基本相同，其中参数 label 是显示在按钮上的文本。

实例 03　为登录界面添加“确认”和“取消”按钮

使用 wx.Button，在实例 02 的基础上添加“确认”和“取消”按钮，具体代码如下：

```
01 # -*- coding:utf-8 -*-
02 import wx
03 class MyFrame(wx.Frame):
04     def __init__(self,parent,id):
05         wx.Frame.__init__(self, parent,id, title="创建TextCtrl类",size=(400, 300))
06         #创建面板
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
07        panel = wx.Panel(self)
08        # 创建文本和密码输入框
09        self.title = wx.StaticText(panel ,label="请输入用户名和密码",pos=
(140,20))
10        self.label_user = wx.StaticText(panel,label="用户名:",pos=(50,50) )
11        self.text_user = wx.TextCtrl(panel,pos=(100,50),size=(235,25),style=
wx.TE_LEFT)
12        self.label_pwd = wx.StaticText(panel,pos=(50,90),label="密  码:")
13        self.text_password = wx.TextCtrl(panel,pos=(100,90),size=(235,25),
style=wx.TE_PASSWORD)
14        #创建“确定”和“取消”按钮
15        self.bt_confirm = wx.Button(panel,label='确定',pos=(105,130))
16        self.bt_cancel = wx.Button(panel,label='取消',pos=(195,130))
17
18 if __name__ == '__main__':
19     app = wx.App()                          # 初始化
20     frame = MyFrame(parent=None,id=-1)  # 实例化 MyFrame 类，并传递参数
21     frame.Show()                            # 显示窗口
22     app.MainLoop()                          # 调用主循环方法
```

运行结果如图 13.8 所示。

图 13.8　添加按钮的登录界面

13.2.6　BoxSizer 布局

在前面的例子中，使用了文本和按钮等控件，并将这些控件通过 pos 参数布置在 pannel 画板上，虽然这种设置坐标的方式很容易理解，但是过程很麻烦。此外，控件的几何位置是绝对位置，也就是固定的。当调整窗口大小时，界面不美观。在 wxPython 中有一种更智能的布局方式——sizer（尺寸器）。sizer 是用于自动布局一组窗口控件的算法。sizer 被附加到一个容器中，通常是一个框架或面板。在父容器中创建的子窗口控件必须被分别添加到 sizer。当 sizer 被附加到容器时，它随后就可以管理它所包含的子布局。

wxPython 提供了 5 个 sizer，如表 13.2 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

表 13.2　wxPython 的 sizer 说明

sizer 名称	描　　述
BoxSizer	在一条水平或垂直线上的窗口部件的布局。当尺寸改变时，控制窗口部件在行为上很灵活。通常用于嵌套的样式。可用于几乎任何类型的布局
GridSizer	一个基础的网格布局。当你要放置的窗口部件都是同样的尺寸且整齐地放入一个规则的网格中可以使用它
FlexGridSizer	对 GridSizer 稍微做了些改变，当窗口部件有不同的尺寸时，可以有更好的结果
GridBagSizer	GridSizer 系列中最灵活的成员。使得网格中的窗口部件可以更随意的放置
StaticBoxSizer	一个标准的 BoxSizer。带有标题和环线

1．什么是 BoxSizer

BoxSizer 是 wxPython 所提供的 sizer 中最简单和最灵活的控件。一个 BoxSizer 是一个垂直列或水平行，窗口部件在其中从左至右或从上到下布置在一条线上。虽然这听起来好像用处不大，但是来自相互之间嵌套 sizer 的能力使你能够在每行或每列很容易放置不同数量的项目。由于每个 sizer 都是一个独立的实体，因此你的布局就有了更多的灵活性。

2．使用 BoxSizer 布局

尺寸器会管理组件的尺寸，只要将部件添加到尺寸器上，再加上一些布局参数，然后让尺寸器自己去管理父组件的尺寸。下面使用 BoxSizer 实现简单的布局，代码如下：

```
01 # -*- coding:utf-8 -*-
02 import wx
03 class MyFrame(wx.Frame):
04     def __init__(self, parent, id):
05         wx.Frame.__init__(self, parent, id, '用户登录', size = (400, 300))
06         #创建面板
07         panel = wx.Panel(self)
08         self.title = wx.StaticText(panel ,label="请输入用户名和密码")
09         #添加容器，容器中控件按纵向排列
10         vsizer = wx.BoxSizer(wx.VERTICAL)
11         vsizer.Add(self.title,proportion=0,flag=wx.BOTTOM|wx.TOP|wx.ALIGN_
CENTER, border = 15 )
12         panel.SetSizer(vsizer)
13 if __name__ == '__main__':
14     app = wx.App()                                  # 初始化
15     frame = MyFrame(parent=None,id=-1)              # 实例化 MyFrame 类，并传递参数
16     frame.Show()                                    # 显示窗口
17     app.MainLoop()                                  # 调用主循环方法
```

运行结果如图 13.9 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 13.9　BoxSizer 基本布局

在上述代码中，设置了增加背景控件（wx.Panel），并创建了一个 wx.BoxSizer，它带有一个决定它是水平还是垂直的参数（wx.HORIZONTAL 或者 wx.VERTICAL），默认为水平。然后使用 Add()方法将控件加入 sizer，最后使用面板的 SetSizer()方法设定它的尺寸器。

Add()方法的语法格式如下：

```
Box.Add(control, proportion, flag, border)
```

参数说明如下。

- control：要添加的控件。
- proportion：所添加控件在定义的定位方式所代表方向上，占据的空间比例。如果有 3 个按钮，它们的比例值分别为 0、1 和 2，它们都已添加到一个宽度为 30 的水平排列的 wx.BoxSizer 中，起始宽度都是 10。当 sizer 的宽度从 30 变成 60 时，按钮 1 的宽度保持不变，仍然是 10，按钮 2 的宽度约为 20[10+(60-30)×1/(1+2)]，按钮 3 的宽度约为 30。
- flag：flag 参数与 border 参数结合使用可以指定边距宽度，包括以下选项。
  - wx.LEFT：左边距。
  - wx.RIGHT：右边距。
  - wx.BOTTOM：底边距。
  - wx.TOP：上边距。
  - wx.ALL：上下左右 4 个边距。

可以通过竖线“|”操作符（operator）来联合使用这些标志，比如 wx.LEFT | wx.BOTTOM。此外，flag 参数还可以与 proportion 参数结合，指定控件本身的对齐（排列）方式，包括以下选项。

- wx.ALIGN_LEFT：左边对齐。
- wx.ALIGN_RIGHT：右边对齐。
- wx.ALIGN_TOP：顶部对齐。
- wx.ALIGN_BOTTOM：底边对齐。
- wx.ALIGN_CENTER_VERTICAL：垂直对齐。
- wx.ALIGN_CENTER_HORIZONTAL：水平对齐。
- wx.ALIGN_CENTER：居中对齐。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- wx.EXPAND：所添加控件将占有 sizer 定位方向上所有可用的空间。

- border：控制所添加控件的边距，就是在控件之间添加一些像素的空白。

实例 04　使用 BoxSizer 设置登录界面布局

使用 BoxSizer 布局方式，实现实例 03 的界面布局效果，具体代码如下：

```
# -*- coding:utf-8 -*-
import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id):
        wx.Frame.__init__(self, parent, id, '用户登录', size=(400, 300))
        # 创建面板
        panel = wx.Panel(self)

        # 创建“确定”和“取消”按钮，并绑定事件
        self.bt_confirm = wx.Button(panel, label='确定')
        self.bt_cancel = wx.Button(panel, label='取消')
        # 创建文本，左对齐
        self.title = wx.StaticText(panel, label="请输入用户名和密码")
        self.label_user = wx.StaticText(panel, label="用户名:")
        self.text_user = wx.TextCtrl(panel, style=wx.TE_LEFT)
        self.label_pwd = wx.StaticText(panel, label="密　码:")
        self.text_password = wx.TextCtrl(panel, style=wx.TE_PASSWORD)
        # 添加容器，容器中控件横向排列
        hsizer_user = wx.BoxSizer(wx.HORIZONTAL)
        hsizer_user.Add(self.label_user, proportion=0, flag=wx.ALL, border=5)
        hsizer_user.Add(self.text_user, proportion=1, flag=wx.ALL, border=5)
        hsizer_pwd = wx.BoxSizer(wx.HORIZONTAL)
        hsizer_pwd.Add(self.label_pwd, proportion=0, flag=wx.ALL, border=5)
        hsizer_pwd.Add(self.text_password, proportion=1, flag=wx.ALL, border=5)
        hsizer_button = wx.BoxSizer(wx.HORIZONTAL)
        hsizer_button.Add(self.bt_confirm, proportion=0, flag=wx.ALIGN_CENTER,
border=5)
       hsizer_button.Add(self.bt_cancel, proportion=0, flag=wx.ALIGN_CENTER,
border=5)
        # 添加容器，容器中控件纵向排列
        vsizer_all = wx.BoxSizer(wx.VERTICAL)
        vsizer_all.Add(self.title, proportion=0, flag=wx.BOTTOM | wx.TOP |
wx.ALIGN_CENTER, border=15)
        vsizer_all.Add(hsizer_user, proportion=0, flag=wx.EXPAND | wx.LEFT |
wx.RIGHT, border=45)
        vsizer_all.Add(hsizer_pwd, proportion=0, flag=wx.EXPAND | wx.LEFT |
wx.RIGHT, border=45)
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
34        vsizer_all.Add(hsizer_button,  proportion=0,  flag=wx.ALIGN_CENTER  |
wx.TOP, border=15)
35        panel.SetSizer(vsizer_all)
36
37  if __name__ == '__main__':
38      app = wx.App()                          # 初始化
39      frame = MyFrame(parent=None,id=-1)  # 实例化 MyFrame 类，并传递参数
40      frame.Show()                            # 显示窗口
41      app.MainLoop()                          # 调用主循环方法
```

在上述代码中，首先创建按钮和文本控件，然后将其添加到容器中，并且设置横向排列。接着，设置纵向排列。在布局的过程中，通过设置每个控件的 flag 和 border 参数，实现控件位置间的布局。至此，使用 BoxSizer 将绝对位置布局更改为相对位置布局，运行结果如图 13.10 所示。

图 13.10　使用 BoxSizer 布局登录界面

13.2.7　事件处理

1．什么是事件

完成布局以后，接下来就是输入用户名和密码。当单击“确定”按钮时，检验输入的用户名和密码是否正确，并输出相应的提示信息。当单击“取消”按钮时，清空已经输入的用户名和密码。要实现这样的功能，就需要使用 wxPython 的事件处理。

那么什么是事件呢？用户执行的动作就叫作事件（event），比如单击按钮，就是一个单击事件。

2．绑定事件

当发生一个事件时，需要让程序注意这个事件并且做出反应。这时，可以将函数绑定到所涉及事件可能发生的控件上。当事件发生时，函数就会被调用。利用控件的 Bind()方法可以将事件处理函数绑定到给定的事件上。例如，为“确定”按钮添加一个单击事件，代码如下：

```
bt_confirm.Bind(wx.EVT_BUTTON,OnclickSubmit)
```

参数说明如下。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- wx.EVT_BUTTON：事件类型为按钮类型。在 wxPython 中有很多以 wx.EVT_开头的事件类型，例如，类型 wx.EVT_MOTION 产生于用户移动鼠标。类型 wx.ENTER_WINDOW 和 wx.LEAVE_WINDOW 产生于当鼠标进入或离开一个窗口控件。类型 wx.EVT_MOUSEWHEEL 被绑定到鼠标滚轮的活动事件中。
- OnclickSubmit：方法名。事件发生时执行该方法。

实例 05　使用事件判断用户登录

在实例 04 的基础上，分别为“确定”和“取消”按钮添加单击事件。在用户输入用户名和密码后，单击“确定”按钮，如果输入的用户名为“mr”并且密码为“mrsoft”，则弹出对话框提示“登录成功”，否则提示“用户名和密码不匹配”。当用户单击“取消”按钮时，清空用户输入的用户名和密码。关键代码如下：

```
# -*- coding:utf-8 -*-
import wx

class MyFrame(wx.Frame):
    def __init__(self, parent, id):
        wx.Frame.__init__(self, parent, id, '用户登录', size=(400, 300))
        # 创建面板
        panel = wx.Panel(self)

        # 创建“确定”和“取消”按钮，并绑定事件
        self.bt_confirm = wx.Button(panel, label='确定')
        self.bt_confirm.Bind(wx.EVT_BUTTON,self.OnclickSubmit)
        self.bt_cancel = wx.Button(panel, label='取消')
        self.bt_cancel.Bind(wx.EVT_BUTTON,self.OnclickCancel)
        # 省略其余代码

    def OnclickSubmit(self,event):
        """ 单击确定按钮，执行方法 """
        message = ""
        username = self.text_user.GetValue()          # 获取输入的用户名
        password = self.text_password.GetValue()      # 获取输入的密码
        if username == "" or password == "" :         # 判断用户名或密码是否为空
            message = '用户名或密码不能为空'
        elif username =='mr' and password =='mrsoft':   # 用户名和密码正确
            message = '登录成功'
        else:
            message = '用户名和密码不匹配'              # 用户名或密码错误
        wx.MessageBox(message)                        # 弹出提示框

    def OnclickCancel(self,event):
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
31          """ 单击取消按钮，执行方法 """
32          self.text_user.SetValue("")                 # 清空输入的用户名
33          self.text_password.SetValue("")             # 清空输入的密码
34
35  if __name__ == '__main__':
36      app = wx.App()                                  # 初始化应用
37      frame = MyFrame(parent=None, id=-1)             # 实例化 MyFrame 类，并传递参数
38      frame.Show()                                    # 显示窗口
39      app.MainLoop()                                  # 调用主循环方法
```

在上述代码中，分别使用 bind()函数为 bt_confirm 和 bt_cancel 绑定了单击事件，单击“确定”按钮时，执行 OnclickSubmit()方法判断用户名和密码是否正确，然后使用 wx.MessageBox()弹出提示框。单击“取消”按钮时，执行 OnclickCancel()方法。用户名和密码正确运行结果如图 13.11 所示，用户名或密码错误的运行结果如图 13.12 所示。

图 13.11　用户名和密码正确

图 13.12　用户名或密码错误

13.3　PyQt框架的使用

Python 起初是一门开发脚本的语言，并不支持 GUI 的功能，几乎都是通过控制台来运行程序的。但是由于 Python 具有非常好的扩展性能，因此现在已经有很多的 GUI 模块可以在 Python 中使用了。自从 Qt 被移植到 Python 中当作框架使用后，已经发布了多个版本，包括 PyQt3、PyQt4 以及目前的 PyQt5 三个版本。

13.3.1　安装 PyQt

Qt 是 Python 开发窗体的工具之一，它不仅与 Python 有着良好的兼容性，还可以通过可视化拖曳的方式进行窗体的创建，提高开发人员的开发效率，因此受到开发人员的喜爱。QT 工具分别支持 Windows、Linux、Mac OS X 三种操作系统，在 https://www.qt.io/download 官方网站中下载对应的系统版本即可。

Qt 工具安装完成以后，还需要在 Python 中安装 PyQt5 模块。PyQt5 模块有两种安装方式，一种是直接在 PyQt5 的官方网站（https://www.riverbankcomputing.com）中下载最新的源码进

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

行编译安装，还可以使用 pip install 的方式进行在线安装，推荐使用后者。使用 pip install pyqt5 安装方式如图 13.13 所示。

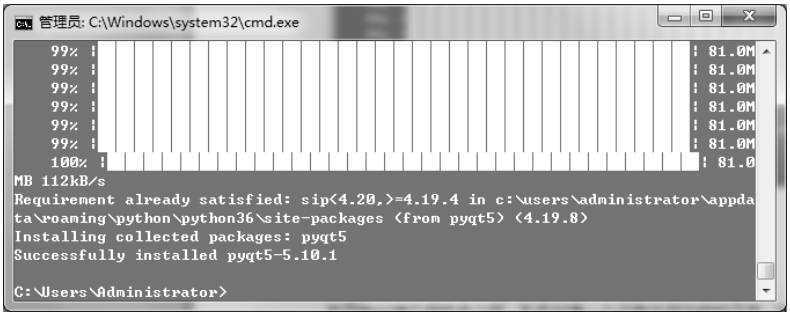


图 13.13　安装 PyQt5 模块

13.3.2　使用第三方开发工具

由于 Qt 在创建窗体项目时会自动生成后缀名为 ui 的文件，该文件需要转换为 py 文件后才可以被 Python 所识别，所以需要为 Qt 与 PyCharm 开发工具进行配置，具体步骤如下。

（1）确保 Python、Qt 与 PyCharm 开发工具安装成功后，打开 PyCharm 开发工具，在欢迎界面中依次单击“Configure”→“Settings”，如图 13.14 所示。

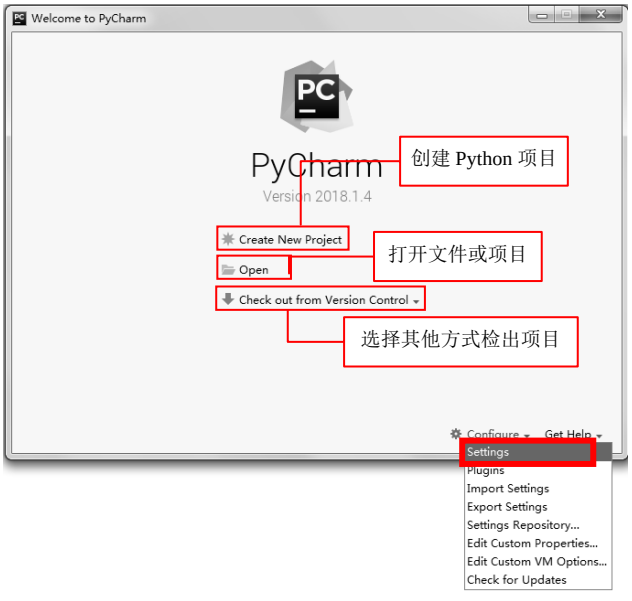


图 13.14　打开 PyCharm 工具的设置界面

（2）打开设置界面后，首先选择“Project Interpreter”选项，然后在右侧的列表中选择“Show All…”，如图 13.15 所示。然后在弹出的对话框中单击 + （添加）按钮，如图 13.16 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

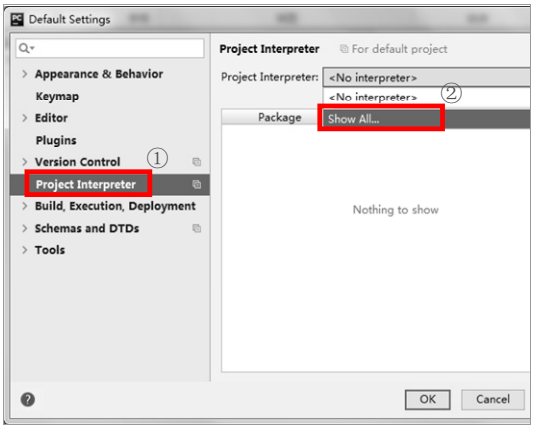


图 13.15　设置界面

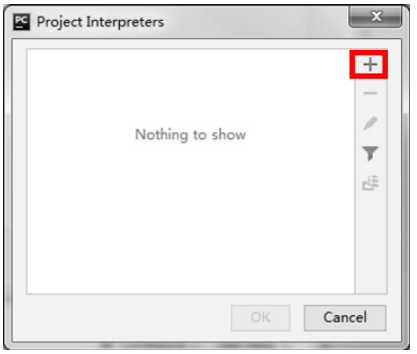


图 13.16　添加 Python 编译版本

（3）在弹出的对话框中选择“System Interpreter”，然后在右侧的下拉列表中默认选择 Python 对应版本的安装路径，单击“OK”按钮即可，如图 13.17 所示。然后在返回的对话框中直接单击“OK”按钮即可，如图 13.18 所示。

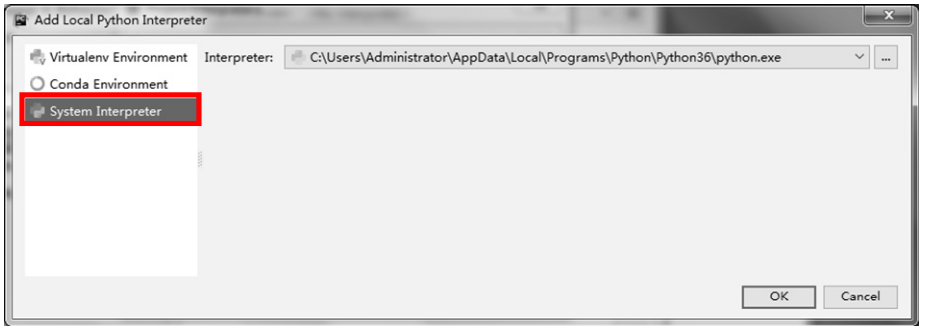


图 13.17　选择 Python 的编译版本

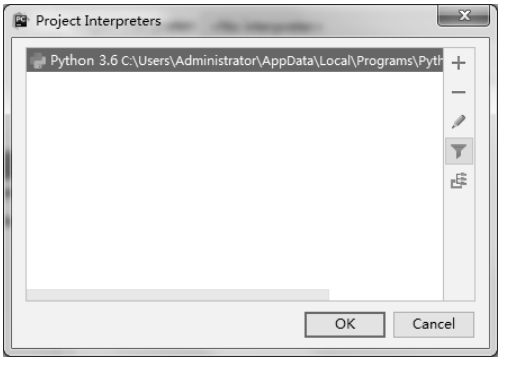


图 13.18　确认 Python 的编译版本

（4）确认了 Python 的编译版本后，在返回的对话框中单击右侧的添加按钮，如图 13.19 所示。然后在弹出的对话框口中添加 PyQt5 模块包，单击“Install Package”按钮，如图 13.20 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

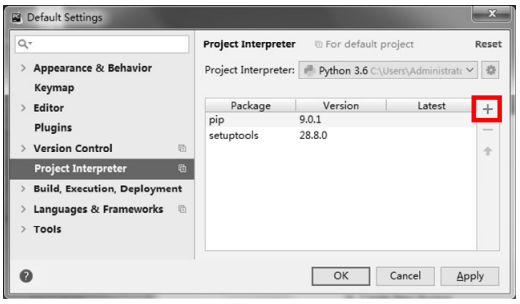


图 13.19　单击添加按钮

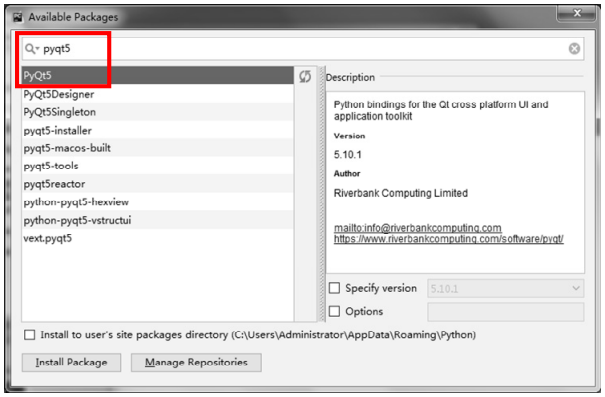


图 13.20　安装 PyQt5 模块包

（5）PyQt5 模块包安装完成后返回到如图 13.21 所示的设置对话框，在该对话框中依次单击“Tools”→“External Tools”选项，然后在右侧单击添加按钮，如图 13.22 所示。

（6）在弹出的对话框中添加启动 Qt Designer 的快捷工具，首先在“Name”编辑框中填写工具名称为“Qt Designer”，然后在“Program”编辑框中填写 Qt 开发工具的安装路径，最后在“Working directory”编辑框中填写“$ProjectFileDir$”，该值是项目文件的目录，单击“OK”按钮，如图 13.23 所示。

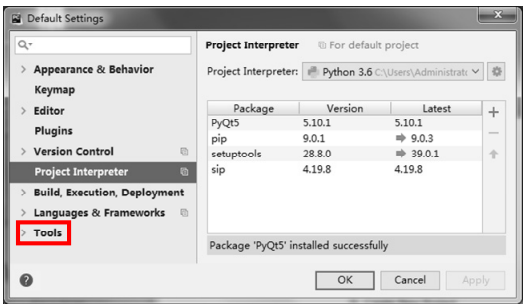


图 13.21　返回设置对话框

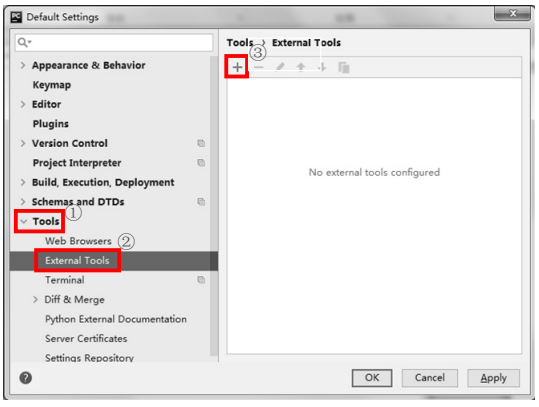


图 13.22　添加外部工具

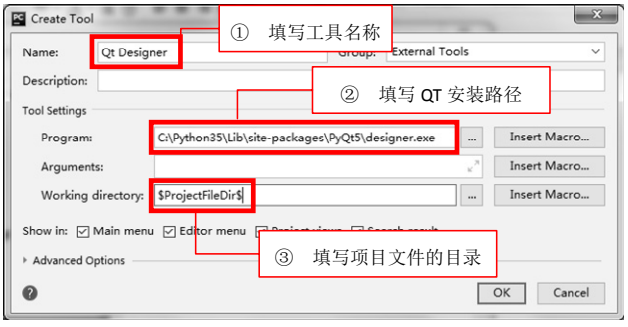


图 13.23　添加启动 Qt Designer 的快捷工具

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **注意：**在“Program”编辑框中填写自己的 Qt 开发工具安装路径，记得尾部需要填写“designer.exe”。

（7）根据步骤（5）与步骤（6）的操作方法，添加将 Qt 生成的 ui 文件转换为 py 文件的快捷工具，在“Name”编辑框中填写工具名称为 PyUIC，然后在“Program”编辑框中填写 Python 的安装路径，再在“Arguments”编辑框中填写将 ui 文件转换为 py 文件的 Python 代码（-m PyQt5.uic.pyuic　$FileName$ -o $FileNameWithoutExtension$.py），在“Working directory”编辑框中填写“$FileDir$”，该值为文件目录，单击“OK”按钮，如图 13.24 所示。

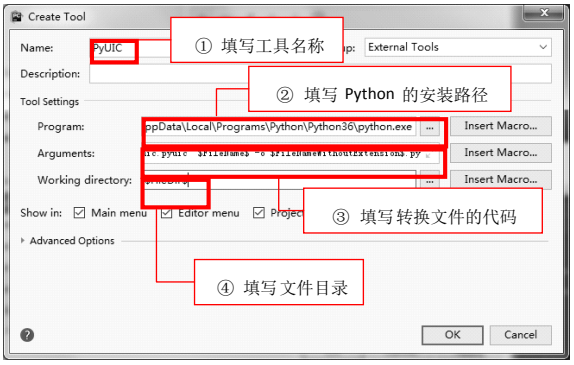


图 13.24　添加将 Qt 生成的 ui 文件转换为 py 文件的快捷工具

> **注意：**在“Program”编辑框中填写自己的 Python 安装路径，记得尾部需要填写“python.exe”。

13.3.3　创建主窗体

由于在上一节中已经将 Python、Qt 与 PyCharm 三个开发工具进行了环境配置，所以创建窗体时只需要启动 PyCharm 开发工具，然后在顶部的菜单栏中依次选择“Tools”→“External Tools”→“Qt Designer”命令，如图 13.25 所示。

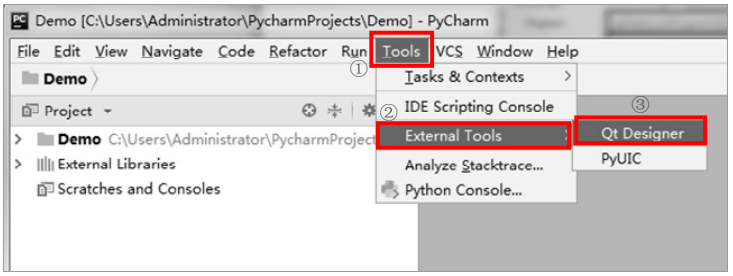


图 13.25　启动 Qt Designer 工具

> **说明：**Qt Designer 是 Qt 工具中的设计师，通过可视化的方式进行程序窗体的设计。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

单击 Qt Designer 快捷工具后，Qt 的窗口编辑工具将自动打开，并且会自动弹出一个新建窗体的对话框，在该对话框中选择一个主窗体的模板，这里选择“Main Window”，然后单击“创建”按钮即可，如图 13.26 所示。

图 13.26　选择主窗体模板

主窗体创建完成后，自动进入 Qt Designer 的设计界面，顶部区域是菜单栏与菜单快捷选项，左侧区域是各种控件与布局，中间的区域为编辑区域，可以将控件拖曳至该区域，也可以预览窗体的设计效果。右侧上方是对象查看器，此处列出所有控件以及彼此所属的关系层。右侧中间的位置是属性编辑器，此处可以设置控件的各种属性。右侧底部的位置分别为信号/槽编辑器、动作编辑器及资源浏览器，具体位置与功能如图 13.27 所示。

图 13.27　Qt Designer 的设计界面

向下拖动左侧的控件与布局列表，然后向编辑区域的主窗体中拖入 1 个 Lable 控件，修改需要显示的文字，如图 13.28 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

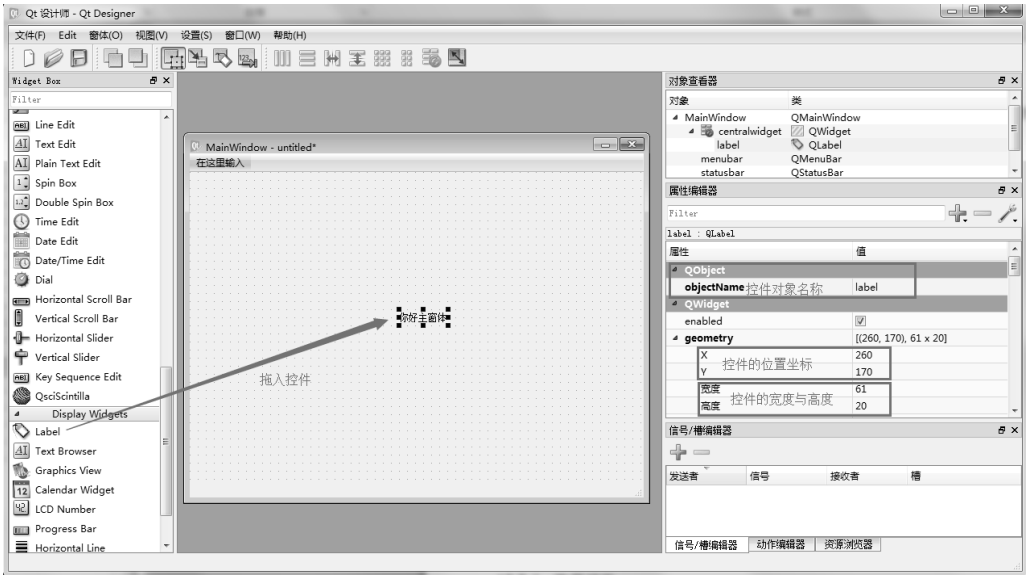


图 13.28　向主窗体拖入控件

按快捷键<Ctrl+S>保存已经创建的主窗体文件，并将该文件名称修改为“Window.ui”，然后使用 PyCharm 开发工具创建 1 个 Demo 项目，再将 Window.ui 文件复制到该项目当中，用鼠标左键选中 Window.ui 文件，然后单击鼠标右键，在弹出的菜单中依次选择“External Tools”→“PyUIC”命令，将在 Demo 项目的目录中自动添加 Window.py 文件，如图 13.29 所示。

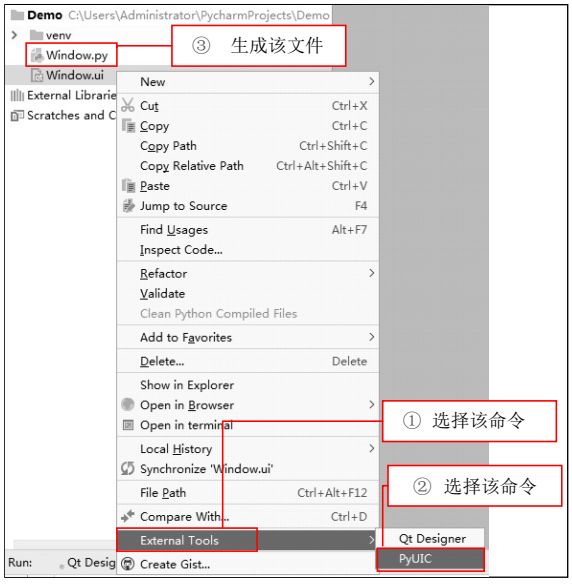


图 13.29　添加 Window.py 文件

打开 Window.py 文件，导入 sys 模块，然后在代码块的最外层创建 show_MainWindow() 方法，该方法用于显示窗体。关键代码如下：

```
01 import sys  # 导入系统模块
02 def show_MainWindow():
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
03     app = QtWidgets.QApplication(sys.argv) # 实例化 QApplication 类，作为 GUI 主
程序入口
04     MainWindow = QtWidgets.QMainWindow()    # 创建 MainWindow
05     ui = Ui_MainWindow()          # 实例化 UI 类
06     ui.setupUi(MainWindow)        # 设置窗体 UI
07     MainWindow.show()             # 显示窗体
08     sys.exit(app.exec_())         # 当窗口创建完成时，需要结束主循环过程
```

在代码块的最外层模拟 Python 的程序入口，然后调用显示窗体的 show_MainWindow()方法。关键代码如下：

```
01 if __name__ == "__main__":
02    show_MainWindow()
```

在该文件的右键快捷菜单中选择“Run 'Window'”命令将显示主窗体界面，如图 13.30 所示。

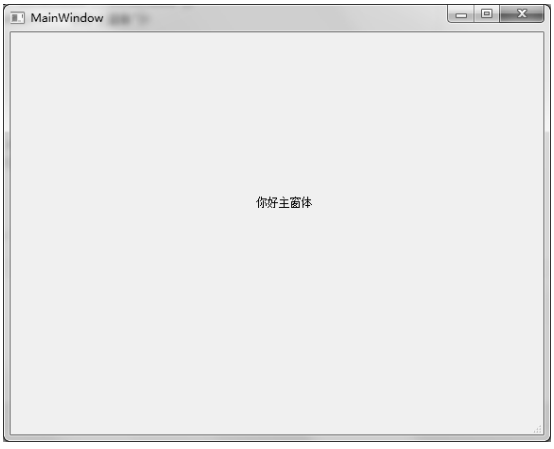


图 13.30　显示主窗体

> **说明：**Lable 控件可以作为一个占位符，显示不可编辑的文本或图片，如果将.ui 文件转换为.py 文件，Lable 控件所对应的类为 QLabel，其他控件也是如此。

13.3.4　常用控件

1．QLineEdit 文本框

QLineEdit 是单行文本框，该控件只能输入单行字符串。QLineEdit 控件还有 1 个兄弟是 QTextEdit 控件，它是多行文本框，可以输入多行字符串。

QLineEdit 的常用方法如表 13.3 所示。

表 13.3　QLineEdit 的常用方法

方法名称	描　　述
setText()	设置文本框内显示的内容
text()	获取文本框内容
setPlaceholderText()	设置文本框浮显文字

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

方法名称	描　　述
setMaxLength()	设置允许文本框内输入字符的最大长度
setEchoMode()	设置文本框显示字符的模式，有以下 4 种模式： • QLineEdit.Normal，显示输入的字符，这是默认设置 • QLineEdit.NoEcho，不显示任何所输入的字符 • QLineEdit.Password，显示与平台相关的密码掩码字符，而不是实际输入的字符 • QLineEdit.PasswordEchoOnEdit，在编辑时显示字符，失去焦点后显示密码掩码字符
clear()	清除文本框内容

实例 06　QLineEdit 控件实现登录界面

使用 QLabel 与 QLineEdit 单行文本框控件，实现一个包含用户名和密码的登录界面。具体步骤如下。

（1）打开 Qt Designer 工具，根据需求，在控件与布局的列表中向主窗体拖入两个 Label 控件与两个 LineEdit 控件，然后为 Label 控件修改需要显示的文字，如图 13.31 所示。

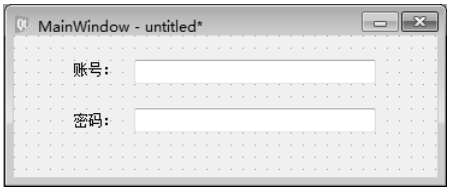


图 13.31　控件摆放位置

（2）选中主窗体的空白处，然后在右侧的属性编辑器中找到 windowTitle 属性，并将标题名称修改为“QLineEdit 单行文本框”，如图 13.32 所示。

图 13.32　修改主窗体的标题名称

（3）分别选中显示“账号”与“密码”的 Label 控件，然后在右侧的属性编辑器中找到 font 属性，并将“点大小（字体大小）”的值修改为“12”，如图 13.33 所示。

图 13.33　修改字体大小

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（4）分别选中需要输入“账号”与“密码”的 QLineEdit 单行文本框，然后在右侧的属性编辑器中，找到 placeholderText 属性，并将值（浮显文字）修改为“请输入账号”与“请输入密码”，如图 13.34 所示。

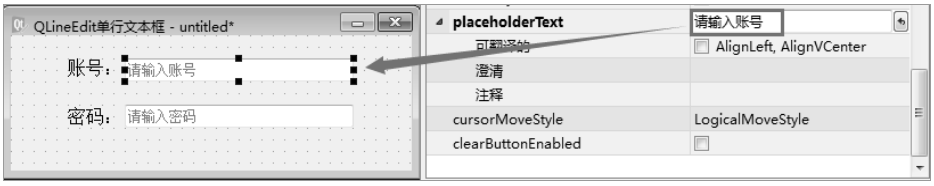


图 13.34　为 QLineEdit 控件设置浮显文字

（5）选中输入密码的 QLineEdit 控件，然后在右侧的属性编辑器中，找到 echoMode 属性，并将值（显示字符的模式）修改为“Password”，如图 13.35 所示。

图 13.35　设置输入密码文本框所显示字符的模式

（6）保存设计窗体的 ui 文件，然后将.ui 文件转换为.py 文件，导入 sys 模块，再添加显示主窗体的 show_MainWindow()方法，最后在程序的入口调用该方法，运行程序后，主窗体默认显示如图 13.36 所示，然后分别输入账号与密码后将显示如图 13.37 所示的效果。

图 13.36　主窗体默认效果

图 13.37　输入账号与密码后的效果

QTextEdit 多行文本框控件，可以显示多行的文本内容，当文本内容超出控件的显示范围时，该控件将显示垂直滚动条。QTextEdit 控件不仅可以显示文本内容，还可以显示 HTML 文档信息。

QTextEdit 的常用方法如表 13.4 所示。

表 13.4　QTextEdit 的常用方法

方法名称	描　　述
setPlainText()	设置文本内容
toPlainText()	获取文本内容
setTextColor()	设置文本颜色，例如，红色可以将参数设置为 QtGui.QColor(255,0,0)
setTextBackgroundColor()	设置文本的背景颜色，颜色参数与 setTextColor()相同
setHtml()	设置 HTML 文档内容
toHtml()	获取 HTML 文档内容
wordWrapMode()	设置自动换行
clear()	清除所有内容

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2．QPushButton 按钮控件

QPushButton 是 PyQt 中最普通也是最常用的按钮之一，QPushButton 的常用方法如表 13.5 所示。

表 13.5　QPushButton 的常用方法

方法名称	描　　述
setText()	设置按钮所显示的文本
text()	获取按钮所显示的文本
setIcon()	设置按钮上的图标，可以将参数设置为 QtGui.QIcon('图标路径')
setIconSize()	设置按钮图标的大小，参数可以设置为 QtCore.QSize(int width,int height)
setEnabled()	设置按钮是否可用，参数设置为 False 时，按钮为不可用状态
setShortcut()	设置按钮的快捷键，参数可以设置为键盘中的按键或组合键，例如'Alt+0'

如果需要 QPushButton 控件实现单击效果的时候，可以使用以下的代码：

```
# 参数中的 self.click 为单击事件所触发的方法名称
self.pushButton.clicked.connect(self.click)
```

3．QRadioButton 单选按钮

QRadioButton 也是按钮的一种，多数用于实现“二选一”或“多选一”的选择现象。QRadioButton 的常用方法如表 13.6 所示。

表 13.6　QRadioButton 的常用方法

方法名称	描　　述
setText()	设置单选按钮显示的文本
text()	获取单选按钮显示的文本
setChecked()	设置单选按钮是否为选中状态，True 为选中状态
isChecked()	返回单选按钮的状态，True 为选中状态，False 为未选中状态

如果需要监测单选按钮的选中状态时，可以使用如下代码：

```
    # 设置单选按钮的选中事件方法
    self.radioButton.toggled.connect(lambda :self.button_state(self.
radioButton))
    self.radioButton_2.toggled.connect(lambda :self.button_state(self.
radioButton_2))

def button_state(self,button):
    if button.text()=='RadioButton1':       # 判断单选按钮的名称
        if button.isChecked() == True:      # 判断单选按钮是否被选中
            print(button.text()+'已选中！')
        else:
            print(button.text()+'未选中！')

    if button.text()=='RadioButton2':       # 判断单选按钮的名称
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
13          if button.isChecked() == True:       # 判断单选按钮是否被选中
14              print(button.text()+'已选中! ')
15          else:
16              print(button.text()+'未选中! ')
```

13.3.5　布局管理

Qt Designer 工具提供了 4 种布局方式，分别为 Vertical Layout（垂直布局）、Horizontal Layout（水平布局）、Grid Layout（网格布局）以及 Form Layout（表单布局），它们都位于 Qt Designer 工具的左侧列表中，如图 13.38 所示。

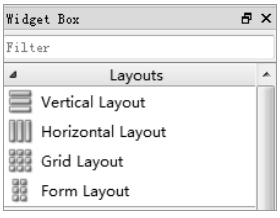


图 13.38　布局列表

- 垂直布局：按照从上至下的顺序显示控件。
- 水平布局：按照从左至右的顺序显示控件。
- 网格布局：将控件放入网格中，然后将控件合理地分成若干个行（row）与列（column），再将每一个控件放置在合适的单元（cell）中。
- 表单布局：控件是以两列的方式布局在表单中，左列包含标签，右列包含输入控件。

1．通过布局管理器布局

打开 Qt Designer 工具，从左侧的控件列表中向主窗体拖入 Label 与 LineEdit 控件，然后按住 Ctrl 键用鼠标左键选中以上拖入的两个控件，选中后单击鼠标右键，在弹出的快捷菜单中依次选择“布局”→“水平布局”命令，如图 13.39 所示。

图 13.39　选择控件的布局方式

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

选择了水平布局以后，控件在主窗体中将自动按照从左至右的顺序显示控件，如果需要调整控件的位置，选中控件拖至目标位置即可，如图 13.40 所示。

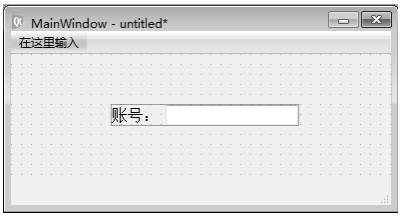


图 13.40　水平布局控件位置

2．绝对布局

最简单的布局方式就是绝对布局，就是在 Qt Designer 工具中右侧的属性编辑器中，通过 geometry 属性来设置控件的位置与大小，以图 13.40 中的 Label 控件为例，该控件的 geometry 属性值如图 13.41 所示。

geometry	[(100, 51), 48 x 16]
X	100
Y	51
宽度	48
高度	16

图 13.41　设置控件的 geometry 属性

在图 13.41 中，X 对应的值是 100，表示以控件左上角为原点，横向距离主窗体左边框 100px（像素）。Y 对应的值是 51，表示以控件左上角为原点，纵向距离主窗体顶部边框 51px（像素），如图 13.42 所示。

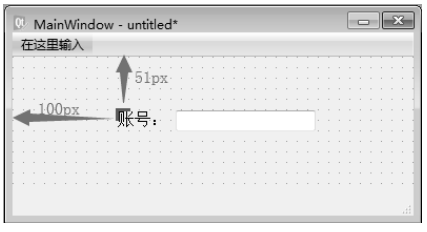


图 13.42　控件的原点与位置

说明：图 13.41 中“宽度”值对应的是控件的宽度，“高度”值对应的是控件的高度。

13.3.6　信号与槽的关联

信号（signal）/槽（slot）是 Qt 中非常重要的一部分，通过信号和槽的关联就可以实现对象之间的通信。当信号发射（emit）时，连接的槽函数（方法）将会自动执行。

1．编辑信号/槽

例如，通过信号/槽实现一个单击按钮关闭主窗体的运行效果，具体操作步骤如下。

（1）打开 Qt Designer 工具，然后在左侧的列表中找到 PushButton 控件，将该控件拖入主

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

窗体中，然后修改需要显示的文字“单击关闭窗体”，如图 13.43 所示。

图 13.43　拖入关闭窗体的按钮

（2）在顶部工具栏中，单击“编辑信号/槽”按钮，然后用鼠标左键按住“单击关闭窗体”按钮，拖动至接收者主窗体，如图 13.44 所示。

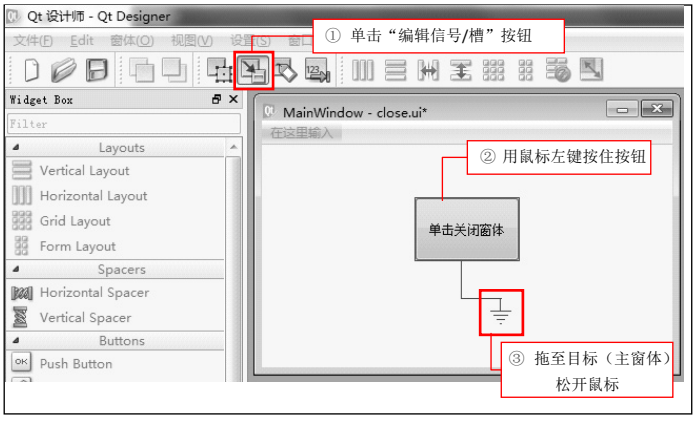


图 13.44　编辑信号/槽

（3）拖至主窗体松开鼠标后，将自动弹出“配置连接”对话框，勾选“显示从 QWidget 继承的信号和槽”复选框，再一次选中“clicked()”→“close()”，单击“OK”按钮，如图 13.45 所示。

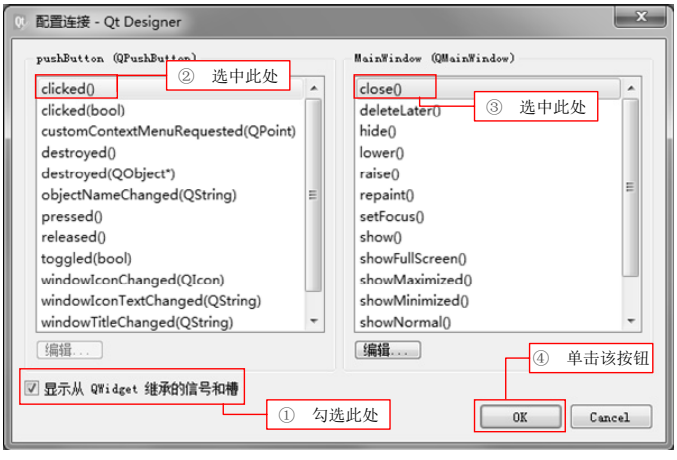


图 13.45　设置信息与槽

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **说明：**在步骤（3）中，选中的 clicked()为按钮的信号，然后选中的 close()为槽函数（方法），工作逻辑是，单击按钮时发射 clicked 信号，该信号被主窗体的槽函数（方法）close()所捕获，并触发了关闭主窗体的行为。

将.ui 文件保存，然后转换为.py 文件，转换后实现单击按钮关闭窗体的关键代码如下：

```
self.pushButton.clicked.connect(MainWindow.close)
```

2．信号/槽编辑器

除了在顶部工具栏中选中“编辑信号/槽”的选项以外，还可以在右下角的“信号/槽编辑器”中进行设置。在“信号/槽编辑器”中单击（添加）按钮，以上一节实现的效果为例，双击“发送者”，选择“pushButton”控件，然后在“信号”中选择“clicked()”，在“接收者”中选择“MainWindow”，最后在“槽”中选择“close()”，如图 13.46 所示。

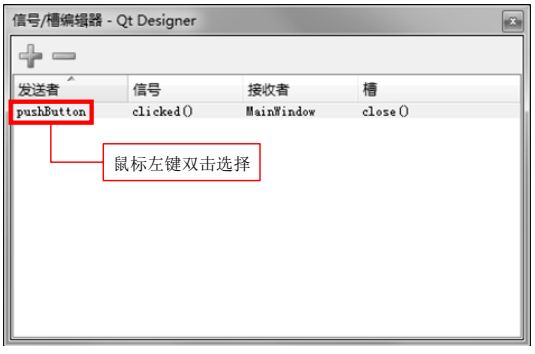


图 13.46　信号/槽编辑器

将.ui 文件保存，然后转换为.py 文件，转换后实现单击按钮关闭窗体的关键代码如下：

```
self.pushButton.clicked.connect(MainWindow.close)
```

13.3.7　资源文件的使用

1．Qt Designer 加载资源文件

在 Qt Designer 工具中设计程序界面时，不可以直接使用图片和图标等资源，而是需要通过资源浏览器添加图片或图标等资源，具体步骤如下。

（1）在 Python 的项目路径中创建一个名称为“images”的文件夹，然后将需要测试的图片保存在该文件夹中，打开 Qt Designer 工具，在右下角的资源浏览器中单击“编辑资源”按钮，如图 13.47 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

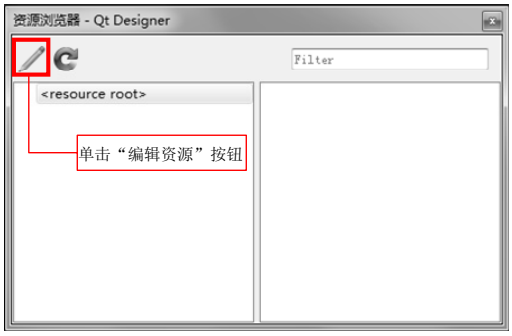


图 13.47　单击“编辑资源”按钮

（2）在弹出的“编辑资源”对话框中单击左下角的第一个按钮“新建资源文件”，如图 13.48 所示。

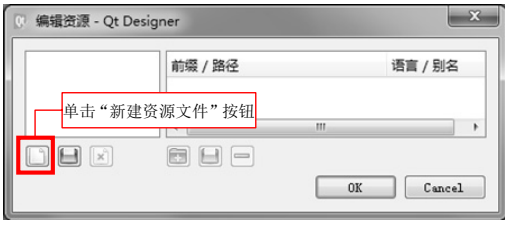


图 13.48　单击“新建资源文件”按钮

（3）在“新建资源文件”对话框中，首先选择该资源文件保存的路径为当前 Python 的项目路径，然后设置文件名称为“img”，保存类型为“资源文件（*.qrc）”，最后单击“保存”按钮，如图 13.49 所示。

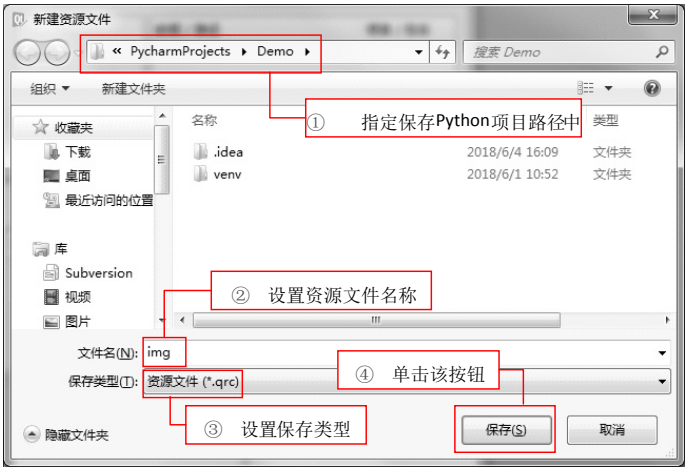


图 13.49　新建资源文件

（4）单击“保存”按钮后，将自动返回至“编辑资源”对话框中。在该对话框中单击“添加前缀”按钮，设置前缀为“png”，再单击“添加文件”按钮，如图 13.50 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 13.50 单击“添加前缀”按钮

（5）在弹出的“添加文件”对话框中选择需要添加的图片文件，然后单击“打开”按钮即可，如图 13.51 所示。

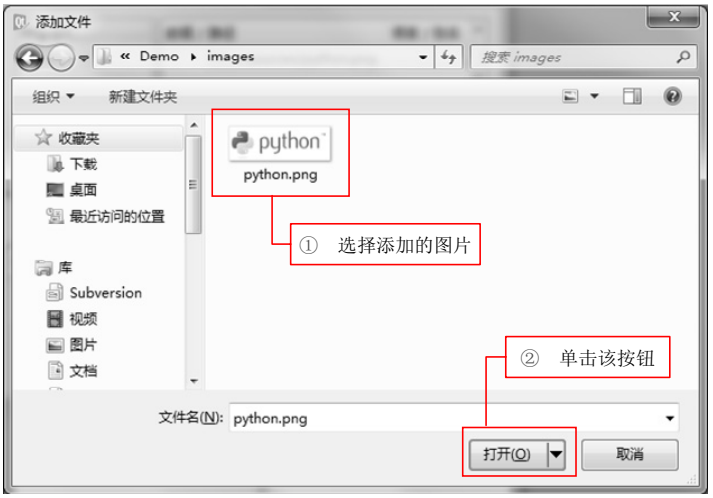


图 13.51 选择添加的图片

（6）图片添加完成以后，将自动返回至“编辑资源”对话框，在该对话框中直接单击“OK”按钮即可，资源浏览器将显示添加的图片资源，效果如图 13.52 所示。

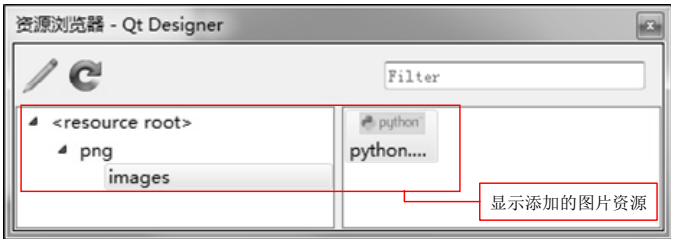


图 13.52 显示添加的图片资源

说明：设置的前缀是我们自己定义的路径前缀，用于区分不同的资源文件。

（7）创建主窗体，然后向主窗体中拖入一个 Label 控件，将控件的大小尺寸设置为与图片相同，然后找到 pixmap 属性，在右侧值的位置选择刚刚创建的图片资源，如图 13.53 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

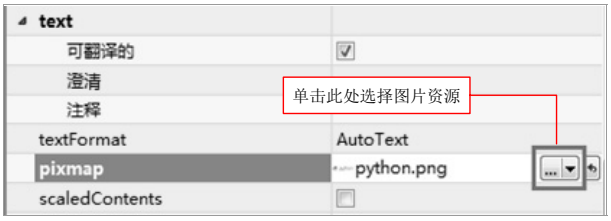


图 13.53　选择图片资源

（8）图片资源选择完成以后，主窗体中的 Label 控件将显示如图 13.54 所示的效果。

图 13.54　Label 控件显示指定的图片资源

2．资源文件的转换

在 Qt Designer 工具中已经了解了如何创建图片资源与图片资源的使用，然后将已经设计好的.ui 文件转换为.py 文件，转换后代码中将显示如图 13.55 所示的提示信息。

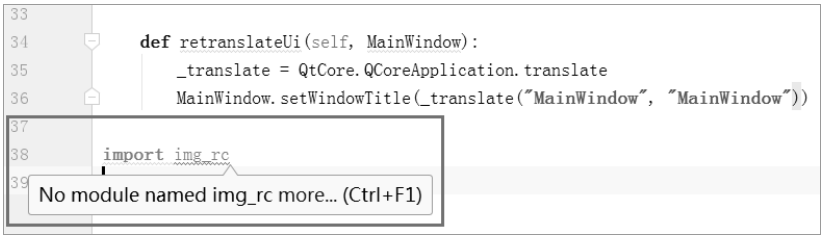


图 13.55　转换后的.py 文件

图 13.55 中的提示信息说明 img_rc 模块导入出现异常，所以此处需要将已经创建好的 img.qrc 资源文件转换为.py 文件，这样主窗体的.py 文件才可以正常显示，资源文件转换的具体步骤如下。

（1）在 PyCharm 开发工具中添加将.qrc 文件转换为.py 文件的快捷工具，在“Name”编辑框中填写工具名称为“qrcTopy”，然后在“Program”编辑框中填写 pyrcc.exe 的安装路径，再在“Arguments”编辑框中填写转换代码（$FileName$ -o $FileNameWithoutExtension$_rc.py），在“Working directory”编辑框中填写“$FileDir$”，该值为文件目录，单击“OK”按钮即可，如图 13.56 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

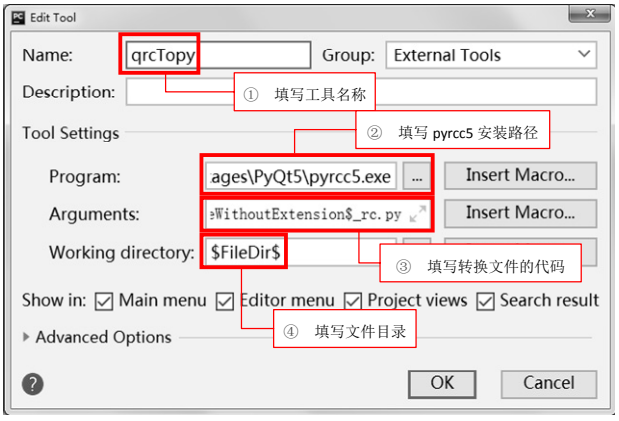


图 13.56　添加将.qrc 文件转换为.py 文件的快捷工具

（2）转换资源文件的快捷工具创建完成以后，用鼠标左键选中需要转换的.qrc 文件，然后在顶部的菜单栏中依次选择“Tools”→“External Tools”→“qrcTopy”命令，此时在.qrc 文件的下面会自动生成相对应的.py 文件，如图 13.57 所示。

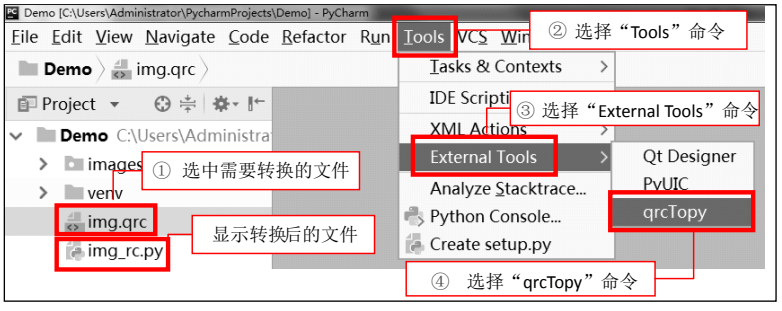


图 13.57　.qrc 转换为.py 文件

（3）文件转换完成以后，图 13.55 中的提示信息将消失。导入 sys 模块，在代码块的最外层创建 show_MainWindow()方法，模拟 Python 的程序入口，然后调用显示窗体的 show_MainWindow()方法。最后运行主窗体文件，主窗体界面如图 13.58 所示。

图 13.58　使用图片资源文件的主窗体

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

13.4　小结

本章主要介绍了 Python 的 GUI 编程，包括 GUI 的基础知识以及 Python 常用的 GUI 框架。在众多 GUI 框架中，我们对 wxPython 与 PyQt 进行了详细讲解。学习使用 wxPython 与 PyQt 创建应用程序，使用常用控件，设置布局以及处理事件等内容。通过本章的学习，读者能够了解 Python 的 GUI 相关知识，并使用 wxPython 与 PyQt 编写交互式的图形界面。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 14 章

Django Web 框架的使用

Django 是基于 Python 的重量级开源 Web 框架。Django 拥有高度定制的 ORM 和大量的 API，简单灵活的视图编写，优雅的 URL，适于快速开发的模板，强大的管理后台，使得它在 Python Web 开发领域占据不可动摇的地位。Instagram、FireFox、国家地理杂志等著名网站都在使用 Django 进行开发。

14.1 Django框架的使用

14.1.1 安装 Django Web 框架

安装 Django Web 框架有 3 种方式，分别是使用 pip 安装 Django、使用 virtualenv 安装 Django 和使用 Anaconda 安装 Django，下面分别进行介绍。

1．使用 pip 安装 Django

在命令行中执行“pip install django==2.0”，即可安装指定的 2.0 版本的 Django 了，如图 14.1 所示。

```
C:\Users\zhang>D:\Webprojects\environments\django2.0\Scripts\activate

(django2.0) C:\Users\zhang>pip install django==2.0
Collecting django==2.0
  Using cached Django-2.0-py3-none-any.whl
Requirement already satisfied: pytz in d:\webprojects\environments\django2.0\lib\site-packages (from django==2.0)
Installing collected packages: django
Successfully installed django-2.0
```

图 14.1　使用 pip 安装 Django

2．使用 virtualenv 安装 Django

在多个项目的复杂工作中，常常会遇到使用不同版本的 Python 包，而虚拟环境则会处理各个包之间的隔离问题。virtualenv 是一种虚拟环境，该环境中可以安装 Django。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

在命令行中执行“pip install virtualenv”命令即可。安装完成后，在命令行中执行“virtualenv D:\Webprojects\venv”即可在 D 盘根目录下的“Webprojects\”文件夹中创建一个名为 venv 的虚拟环境。继续执行“D:\Webprojects\venv\Scripts\activate”命令即可激活虚拟环境。最后在激活后的 venv 中执行“pip install django==1.11.2”命令就可以安装 1.11.2 版本的 Django 了，如图 14.2 所示。

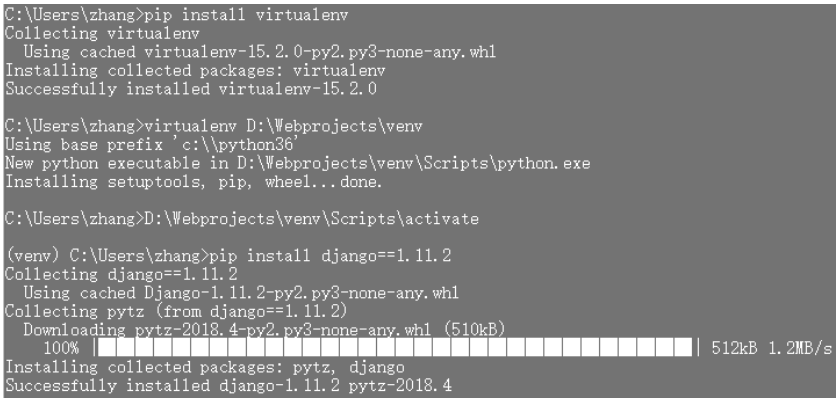


图 14.2　使用 virtualenv 安装 Django

3．使用 Anaconda 安装 Django

Anaconda 也是一种虚拟环境，严格来讲它是一种包管理工具，包含了 conda、Python 等 180 多个科学包及其依赖。Anaconda 的下载地址为 https://www.continuum.io/downloads，下载完成按照提示安装即可。安装完成输入以下命令创建虚拟环境：

```
conda create -n venv1 python=3.6
```

执行上面的命令后，系统会创建一个基于 Python 3.6 的虚拟环境。执行“activate venv1”命令激活环境，然后执行“conda install django”命令安装 Django，如图 14.3 所示。

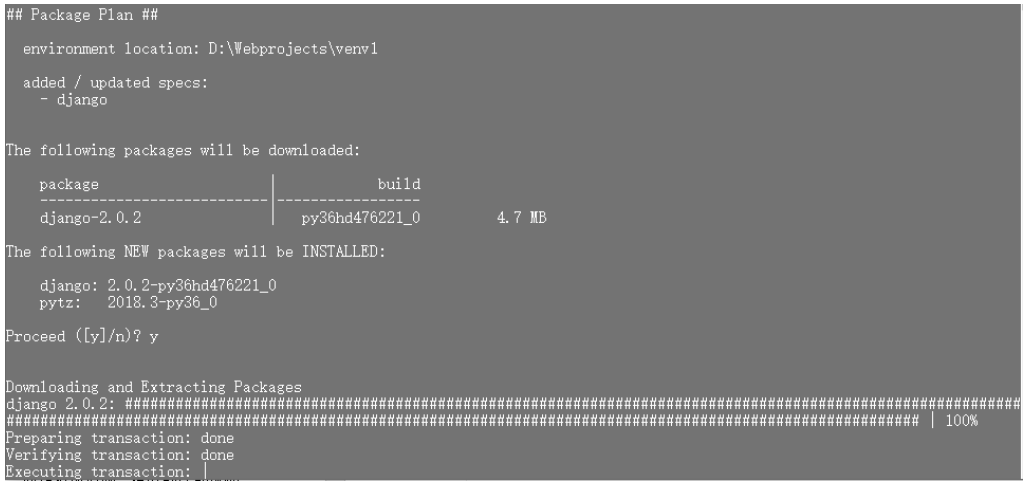


图 14.3　使用 Anaconda 安装 Django

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

14.1.2 创建一个 Django 项目

本节将开始讲解如何使用 Django 创建一个项目，步骤如下。

（1）首先在 D 盘（读者可以根据实际情况选择）根目录下创建用于保存项目文件的目录，这里创建的目录为“D:\Webprojects”。

（2）在 Webprojects 文件夹中创建 environments 目录用于放置虚拟环境，然后在命令行中执行“cmd”命令，输入如下创建环境命令：

```
virtualenv D:\Webprojects\environments\django2.0
```

（3）使用如下命令在命令行激活环境：

```
D:\Webprojects\environments\django2.0\Scripts\activate
```

（4）使用“django-admin”命令创建一个项目：

```
django-admin startproject demo
```

（5）使用 Pycharm 打开 demo 项目，查看目录结构，如图 14.4 所示。

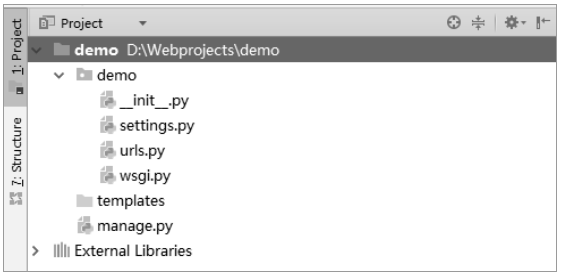


图 14.4　Django 项目目录结构

项目已经创建完成，Django 项目中的文件及说明如表 14.1 所示。

表 14.1　Django 项目中的文件及说明

文　件	说　明
manage.py	Django 程序执行的入口
db.sqlite3	SQLite 的数据库文件，Django 默认使用这种小型数据库存取数据，非必须保留
templates	Django 生成的 HTML 模板文件夹，我们也可以在每个 App 中使用模板文件夹
demo	Django 生成的和项目同名的配置文件夹
settings.py	Django 总的配置文件，可以配置 App、数据库、中间件、模板等诸多选项
urls.py	Django 默认的路由配置文件
wsgi.py	Django 实现的 WSGI 接口的文件，用来处理 Web 请求

（6）在 Pycharm 中单击运行项目，或者在虚拟环境命令行中执行以下命令运行项目：

```
python manage.py runserver
```

此时可以看到 Web 服务器已经开始监听 8000 端口的请求了。在浏览器中输入 http://127.0.0.1:8000，即可看到创建的 Django 项目页面，如图 14.5 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

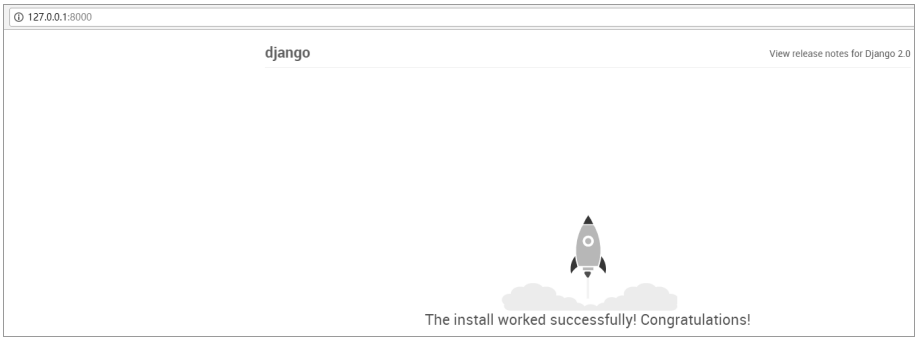


图 14.5　Django 页面

（7）创建完 Django 项目后，在 Pycharm 的命令行执行以下命令，可以为 Django 项目生成数据表，并创建一个账户名和密码。

```
01 python manage.py migrate          # 执行数据库迁移生成数据表
02 python manage.py createsuperuser  # 按照提示输入账户和密码，密码强度符合一定的规则要求
```

效果如图 14.6 所示。

```
(django2.0) D:\Webprojects\demo>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK

(django2.0) D:\Webprojects\demo>python manage.py createsuperuser
Username (leave blank to use 'zhang'): hugo
Email address: demo@demo.org
Password:
Password (again):
Superuser created successfully.
```

图 14.6　为 Django 项目创建账户和密码

（8）重新启动服务器，在浏览器中访问 http://127.0.0.1:8000/admin，使用刚刚创建的账户登录，即可看到后台管理界面，如图 14.7 所示。

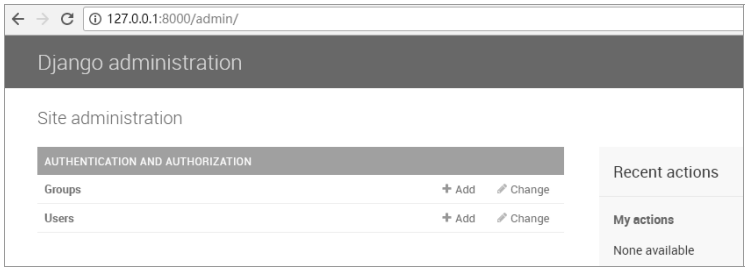


图 14.7　Django 项目后台管理界面

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

14.1.3 创建 App

在 Django 项目中，推荐使用 App 来完成不同模块的任务，通过执行如下命令可以启用一个应用程序：

```
python manage.py startapp app1
```

你会看到项目根目录下又多了一个 app1 的目录，如图 14.8 所示。

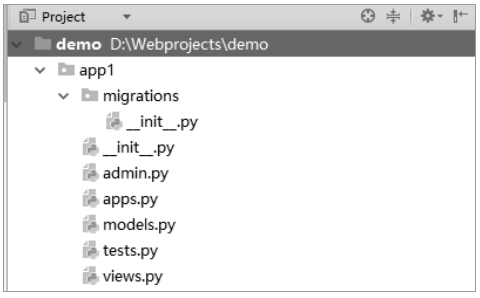


图 14.8　Django 项目的 App 目录结构

Django 项目中 App 目录的文件及说明如表 14.2 所示。

表 14.2　Django 项目中 App 目录的文件及说明

文　件	说　明
migrations	执行数据库迁移生成的脚本
admin.py	配置Django 管理后台的文件
apps.py	单独配置添加的每个App 的文件
models.py	创建数据库数据模型对象的文件
tests.py	用来编写测试脚本的文件
views.py	用来编写视图控制器的文件

下面将已经创建的 App 添加到 settings.py 配置文件中，然后将其激活，否则目录 App 内的文件都不会生效，效果如图 14.9 所示。

```
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      'app1',
41  ]
```

图 14.9　将创建的 App 添加到 settings.py 配置文件中

14.1.4 数据模型（models）

1．在 App 中添加数据模型

在 app1 的 models.py 中添加如下代码:

```
01  from django.db import models  # 引入 django.db.models 模块
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
class Person(models.Model):
    """
    编写 Person 模型类，数据模型应该继承于 models.Model 或其子类
    """
    # 第一个字段使用 models.CharField 类型
    first_name = models.CharField(max_length=30)
    # 第二个字段使用 models.CharField 类型
    last_name = models.CharField(max_length=30)
```

Person 模型中的每一个属性都指明了 models 下面的一个数据类型，代表了数据库中的一个字段。上面的类在数据库中会创建如下的表：

```
CREATE TABLE myapp_person (
    "id" serial NOT NULL PRIMARY KEY,
    "first_name" varchar(30) NOT NULL,
"last_name" varchar(30) NOT NULL
);
```

对于一些公有的字段，为了简化代码，可以使用如下的实现方式：

```
from django.db import models        # 引入 django.db.models 模块
class CreateUpdate(models.Model):  # 创建抽象数据模型，同样要继承于 models.Model
    # 创建时间，使用 models.DateTimeField
    created_at = models.DateTimeField(auto_now_add=True)
    # 修改时间，使用 models.DateTimeField
    updated_at = models.DateTimeField(auto_now=True)
    class Meta:  # 元数据，除了字段以外的所有属性
        # 设置 model 为抽象类。指定该表不应该在数据库中创建
        abstract = True

class Person(CreateUpdate):     # 继承 CreateUpdate 基类
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)

class Order(CreateUpdate):      # 继承 CreateUpdate 基类
    order_id = models.CharField(max_length=30, db_index=True)
    order_desc = models.CharField(max_length=120)
```

我们用于创建日期和修改日期的数据模型就可以继承于 CreateUpdate 类了。上面讲解了数据模型的创建方式，下面介绍 django.db.models 提供的常见字段类型，如表 14.3 所示。

表 14.3　Django 数据模型中常见的字段类型及说明

字段类型	说　明
AutoField	一个 id 自增的字段，但创建表过程 Django 会自动添加一个自增的主键字段
BinaryField	一个保存二进制源数据的字段
BooleanField	一个布尔值的字段，应该指明默认值，管理后台中默认呈现为 CheckBox 形式
NullBooleanField	可以为 None 值的布尔值字段

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

字段类型	说　明
CharField	字符串值字段，必须指明参数 max_length 值。管理后台中默认呈现 TextInput 形式
TextField	文本域字段，对于大量文本应该使用 TextField。管理后台中默认呈现 Textarea 形式
DateField	日期字段，代表 Python 中 datetime.date 的实例。管理后台默认呈现 TextInput 形式
DateTimeField	时间字段，代表 Python 中 datetime.datetime 的实例。管理后台默认呈现 TextInput
EmailField	邮件字段，是 CharField 的实现，用于检查该字段值是否符合邮件地址格式
FileField	上传文件字段。管理后台默认呈现 ClearableFileInput 形式
ImageField	图片上传字段，是 FileField 的实现。管理后台默认呈现 ClearableFileInput 形式
IntegerField	整数值字段。管理后台默认呈现 NumberInput 或者 TextInput 形式
FloatField	浮点数值字段。管理后台默认呈现 NumberInput 或者 TextInput 形式
SlugField	只保存字母、数字、下画线和连接符，用于生成 URL 的短标签
UUIDField	保存一般统一标识符的字段，代表 Python 中 UUID 的实例，建议提供默认值 default
ForeignKey	外键关系字段，需提供外键的模型参数和 on_delete 参数（指定当该模型实例删除的时候，是否删除关联模型），如果要求外键的模型出现在当前模型的后面，需要在第一个参数中使用单引号
ManyToManyField	多对多关系字段，与 ForeignKey 类似
OneToOneField	一对一关系字段，常用于扩展其他模型

2．执行数据库迁移

（1）创建完数据模型后，开始做数据库迁移，首先我们不希望用 Django 默认自带的 SQLite 数据库，我们想使用 MySQL 数据库，在项目的 settings.py 配置文件中找到如下配置：

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

替换为：

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'demo',
        'USER': 'root',
        'PASSWORD': '您的数据库密码'
    }
}
```

（2）创建数据库，在终端连接数据库，执行以下命令：

```
mysql -u root -p
```

（3）按照提示输入数据库密码，连接成功后执行如下语句创建数据库：

```
create database demo default character set utf8;
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

创建成功后，即可在 Django 中使用数据库迁移，并在 MySQL 中创建数据表了。创建数据库命令执行效果如图 14.10 所示。

```
mysql> create database demo default character set utf8;
Query OK, 1 row affected (0.01 sec)
```

图 14.10　创建数据库命令执行效果

（4）安装数据库的驱动，Python 3.x 使用 pymysql 作为 MySQL 的驱动，命令如下：

```
pip install pymysql
```

（5）找到 D:\Webprojcets\demo\demo__init__.py 文件，在行首添加如下代码：

```
01 import pymysql
02 pymysql.install_as_MySQLdb()        # 为了将 pymysql 发挥最大数据库操作性能
```

（6）执行以下命令，用来创建数据表：

```
01 python manage.py makemigrations    # 生成迁移文件
02 python manage.py migrate           # 迁移数据库，创建新表
```

（7）创建数据表的效果如图 14.11 所示。

```
(django2.0) D:\Webprojects\demo>python manage.py makemigrations
Migrations for 'app1':
  app1\migrations\0001_initial.py
    - Create model Order
    - Create model Person

(django2.0) D:\Webprojects\demo>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, app1, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying app1.0001_initial... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying sessions.0001_initial... OK
```

图 14.11　创建数据表

创建完成后，即可在数据库中查看这两张数据表了，Django 会默认按照“App 名称+下画线+模型类名称”的形式创建数据表，对于上面这两个模型，Django 创建了如下表：

- Person 类对应的 app1_person 表。
- Order 类对应的 app1_order 表。

CreateUpdate 是一个抽象类，不会创建表结构，在数据库管理软件中查看创建的数据表，效果如图 14.12 所示。

图 14.12　在数据库管理软件中查看创建的数据表

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

3．了解 Django 数据 API

这里所有的命令将在 Django 的交互命令行中执行，在项目根目录下启用交互命令行，执行以下命令：

```
python manage.py shell  # 启用交互命令行
```

导入数据模型命令如下：

```
from app1.models import Person, Order  # 导入 Person 和 Order 两个类
```

（1）创建数据有如下两种方法。

方法 1：

```
p = Person.objects.create(first_name="hugo", last_name="zhang")
```

方法 2：

```
01  p=Person(first_name="hugo", last_name="张")
02  p.save()  # 必须调用 save()才能写入数据库
```

（2）查询数据。

查询所有数据：

```
Person.objects.all()
```

查询单个数据:

```
Person.objects.get(first_name="hugo")  # 括号内需要加入确定的条件，因为 get 方法只返回一个确定值
```

查询指定条件的数据：

```
01  Person.objects.filter(first_name__exact="hugo") # 指定 first_name 字段值必须为 hugo
02  Person.objects.filter(last_name__iexact="zhang") # 不区分大小写查找值必须为 hugo 的，如 hUgo
03  Person.objects.filter(id__gt=1)    # 查找所有 id 值大于 1 的
04  Person.objects.filter(id__lt=100)  # 查找所有 id 值小于 100 的
05  #排除所有创建时间大于现在时间的，exclude 的用法是排除，和 filter 正好相反
06  Person.objects.exclude(created_at__gt=datetime.datetime.now(tz=datetime.
timezone.utc))
07  #过滤出所有 first_name 字段值包含 h 的，然后将之前的查询结果按照 id 进行排序
08  Person.objects.filter(first_name__contains="h").order_by('id')。
09  Person.objects.filter(first_name__icontains="h") # 查询所有 first_name 值不包含 h 的
```

（3）修改查询到的数据，修改之前需要查询对应的数据或者数据集，代码如下：

```
p = Person.objects.get(first_name="hugo")
```

然后按照需求进行修改，例如：

```
01  p.first_name = "john"
02  p.last_name = "wang"
03  p.save()
```

注意：必须调用 save()方法才能保存到数据库。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

当然也可以使用 get_or_create，如果数据存在就修改，不存在就创建，代码如下：

```
p, is_created = Person.objects.get_or_create(
    first_name="hugo",
    defaults={"last_name": "wang"}
)
```

get_or_create 返回一个元组、一个数据对象和一个布尔值，defaults 参数是一个字典。当获取数据的时候，defaults 参数里面的值不会被传入，也就是说获取的对象只存在 defaults 之外的关键字参数的值。

（4）删除数据同样需要先查找到对应的数据，然后进行删除，代码如下：

```
Person.objects.get(id=1).delete()
(1,({'app1.Person':1}))
```

> **技巧：**在大多数情况下我们不会直接删除数据库中的数据，我们希望在数据模型定义的时候，添加一个 status 字段，值为 True 或 False，用来标记该数据是否为可用状态。在想要删除该数据的时候，将其值设置为 False 即可。

14.1.5　管理后台

定义好数据模型，就可以配置管理后台了，按照如下代码编辑 app1 下面的 admin.py 文件：

```
from django.contrib import admin        # 引入 admin 模块
from app1.models import Person, Order  # 引入数据模型类

class PersonAdmin(admin.ModelAdmin):
    """
    创建 PersonAdmin 类，继承于 admin.ModelAdmin
    """
    # 配置展示列表，在 Person 板块下的列表展示
    list_display = ('first_name', 'last_name')
    # 配置过滤查询字段，在 Person 板块下的右侧过滤框
    list_filter = ('first_name', 'last_name')
    # 配置可以搜索的字段，在 Person 板块下的右侧搜索框
    search_fields = ('first_name',)
    # 配置只读字段展示，设置后该字段不可编辑
    readonly_fields = ('created_at', 'updated_at')
# 绑定 Person 模型到 PersonAdmin 管理后台
admin.site.register(Person, PersonAdmin)
```

配置完成后，启动开发服务器，访问 http://127.0.0.1:8000/admin，效果如图 14.13 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

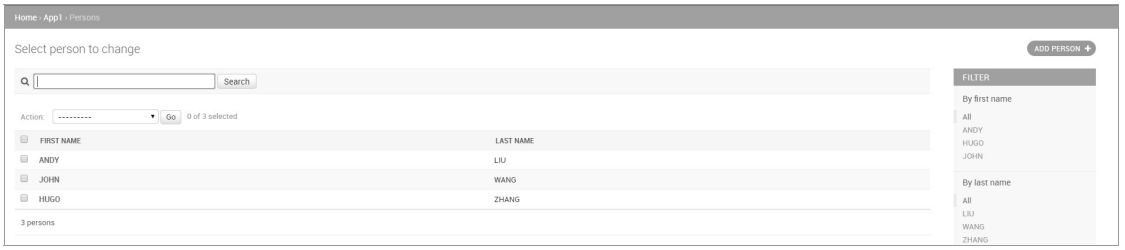


图 14.13　Django 项目后台管理页面

14.1.6　路由（urls）

Django 的 URL 路由流程如下。

（1）Django 查找全局 urlpatterns 变量（urls.py）。

（2）按照先后顺序，对 URL 逐一匹配 urlpatterns 每个元素。

（3）找到第一个匹配时停止查找，根据匹配结果执行对应的处理函数。

（4）如果没有找到匹配或出现异常，Django 进行错误处理。

Django 支持三种表达格式，分别如下。

（1）精确字符串格式，如“articles/2017/”。

一个精确 URL 匹配一个操作函数；最简单的形式，适合对静态 URL 的响应；URL 字符串不以“/”开头，但要以“/”结尾。

（2）Django 的转换格式：<类型:变量名>，如“articles/<int:year>/”。

Django 转换后的格式是一个 URL 模板，匹配 URL 同时在其中获得一批变量作为参数，是一种常用形式，目的是通过 URL 进行参数获取和传递。

表 14.4 提供了一些格式转换类型说明。

表 14.4　格式转换类型说明

格式转换类型	说　明
str	匹配除分隔符（/）外的非空字符，默认类型<year>等价于<str:year>
int	匹配 0 和正整数
slug	匹配字母、数字、横杠、下画线组成的字符串，str 的子集
uuid	匹配格式化的 UUID，如 075194d3-6885-417e-a8a8-6c931e272f00
path	匹配任何非空字符串，包括路径分隔符，是全集

（3）Django 支持的正则表达式格式，如“articles/(?p<year>[0-9]{4})/”。

借助正则表达式的丰富语法表达一类 URL（而不是一个）；可以通过“<>”提取变量作为处理函数的参数；使用该方法时，前面不能使用 path()函数，必须使用 re_path()函数；表达的全部是 str 格式，不能是其他类型。使用正则表达式有两种形式，分别如下。

- 不提取参数：比如“re_path(articles/([0-9]{4})/”，表示 4 位数字，每一个数字都是 0～9 的任意数字。
- 提取参数：命名形式为“(?p<name>pattern)”，比如“re_path(articles/(?p<year>[0-9]{4}))/”，将正则表达式提取的 4 位数字，每一个数字都是 0～9 的任意数字命名为 year。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **注意：**当网站功能较多时，可以在该功能文件夹里创建一个 urls.py 文件，将该功能模块下的 URL 全部写在该文件里，但是要在全局的 urls.py 中使用 include 方法实现 URL 映射分发。

编写 URL 的三种情况如下。

- 普通 URL：re_path('^index/',view.index)，re_path('^home/',view.Home.as_view())。
- 顺序传参：re_path(r'^detail-(\d+)-(\d+).html/',views.detail)。
- 关键字传参：re_path(r'^detail-(?P<nid>\d+)-(?P<uid>\d+).html/',views.detail)。

推荐使用关键字传参的路由方法，找到项目根目录的配置文件夹 demo 下面的 urls.py，打开该文件，并添加如下代码：

```
from django.contrib import admin # 引入默认后台的模块，其中包括管理界面的 urls 路由规则
from django.urls import path, include  # 引入 urls 模块中的 path 方法
urlpatterns = [
    path('admin/', admin.site.urls),
    path('app1/',include('app1.urls'))
]
```

在 app1 下面创建一个 urls.py 文件，并在其中编写属于这个模块的 URL 规则：

```
from app1 import views as app1_views
urlpatterns = [
    # 精确匹配视图
    path('articles/2003/', app1_views.special_case_2003),
    # 匹配一个整数
    path('articles/<int:year>/', app1_views.year_archive),
    # 匹配两个位置的整数
    path('articles/<int:year>/<int:month>/', app1_views.month_archive),
    # 匹配两个位置的整数和一个 slug 类型的字符串
    path('articles/<int:year>/<int:month>/<slug:slug>/', app1_views.article_
detail),
]
```

如果想使用正则表达式匹配，则使用下面的代码：

```

from django.urls import re_path
from app1 import views as views

urlpatterns = [
    # 精确匹配
    path('articles/2003/', views.special_case_2003),
    # 按照正则表达式匹配 4 位数字年份
    re_path(r'^articles/(?P<year>[0-9]{4})/$', views.year_archive),
    # 按照正则表达式匹配 4 位数字年份和 2 位数字月份
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
11     re_path(r'^articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/$',
views.month_archive),
12     # 按照正则表达式匹配 4 位数字年份、2 位数字月份和一个至少 1 位的 slug 类型的字符串
13     re_path(r'^articles/(?P<year>[0-9]{4})/(?P<month>[0-9]{2})/(?P<slug>
[\w-]+)/$', views.article_detail),
14 ]
```

接下来即可通过/app1/articles/2003/12/my_day 访问 app1_views.article_detail 这个视图方法了，效果如图 14.14 所示。

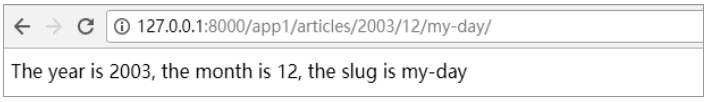


图 14.14　访问视图方法

14.1.7　表单（forms）

在 app1 文件夹下创建一个 forms.py 文件，添加如下类代码：

```
01 from django import forms
02 class PersonForm(forms.Form):
03     first_name = forms.CharField(label='你的名字', max_length=20)
04     last_name = forms.CharField(label='你的姓氏', max_length=20)
```

上面定义了一个 PersonForm 表单类，两个字段类型为 forms.CharField，类似于 models.CharField，first_name 指字段的 label 为你的名字，并且指定该字段最大长度为 20 个字符。max_length 参数可以指定 forms.CharField 的验证长度。

PersonForm 类将呈现为下面的 HTML 代码：

```
01 <label for="你的名字">你的名字: </label>
02 <input id="first_name" type="text" name="first_name" maxlength="20" required />
03 <label for="你的姓氏">你的姓氏: </label>
04 <input id="last_name" type="text" name="last_name" maxlength="20" required />
```

表单类 forms.Form 有一个 is_valid()方法，可以在 views.py 中验证提交的表单是否符合规则。对于提交的内容，在 views.py 中编写如下代码：

```
01 from django.shortcuts import render
02 from django.http import HttpResponse, HttpResponseRedirect
03 from app1.forms import PersonForm
04
05 def get_name(request):
06     # 判断请求方法是否为 POST
07     if request.method == 'POST':
08         # 将请求数据填充到 PersonForm 实例中
09         form = PersonForm(request.POST)
10         # 判断 form 是否为有效表单
11         if form.is_valid():
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            # 使用 form.cleaned_data 获取请求的数据
            first_name = form.cleaned_data['first_name']
            last_name = form.cleaned_data['last_name']
            # 响应拼接后的字符串
            return HttpResponse(first_name + '' + last_name)
        else:
            return HttpResponseRedirect('/error/')
    # 请求为 GET 方法
    else:
        return render(request, 'name.html', {'form': PersonForm()})
```

在 HTML 文件中使用返回的表单的代码如下：

```
<form action="/app1/get_name" method="post"> {% csrf_token %}
    {{ form }}
    <button type="submit">提交</button>
</form>
```

{{form}}是 Django 模板的语法，用来获取页面返回的数据，这个数据是一个 PersonForm 实例，所以 Django 就按照规则渲染表单。

> **注意：**渲染的表单只是表单的字段（如上面 PersonForm）呈现的 HTML 代码，所以我们要在 HTML 中手动输入<form></form>标签，并指出需要提交的路由/app1/get_name 和请求的方法 post。并且，<form>标签的后面需要加上 Django 的防止跨站请求伪造模板标签{% csrf_token %}。一个简单的标签，就很好地解决了 form 表单提交出现跨站请求伪造攻击的情况。

添加 URL 到创建的 app1/urls.py 中，代码如下：

```
path('get_name', app1_views.get_name)
```

此时访问页面 http://127.0.0.1:8000/app1/get_name，效果如图 14.15 所示。

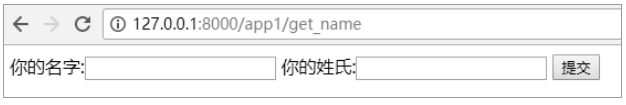


图 14.15　在 Django 项目中创建表单

14.1.8　视图(views)

下面通过一个例子讲解如何在 Django 项目中定义视图，代码如下：

```
from django.http import HttpResponse  # 导入响应对象
import datetime  # 导入时间模块

def current_datetime(request):      # 定义一个视图方法，必须带有请求对象作为参数
    now = datetime.datetime.now()   # 请求的时间
    html = "<html><body>It is now %s.</body></html>" % now  # 生成 HTML 代码
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
07     return HttpResponse(html)  # 将响应对象返回，数据为生成的 HTML 代码
```

上面的代码定义了一个函数，返回了一个 HttpResponse 对象，这就是 Django 的 FBV（Function-Based View）基于函数的视图。每个视图函数都要有一个 HttpRequest 对象作为参数，用来接收来自客户端的请求，并且必须返回一个 HttpResponse 对象，作为响应给客户端。

django.http 模块下有诸多继承于 HttpReponse 的对象，其中大部分在开发中都可以用到。例如，我们想在查询不到数据时，给客户端一个 HTTP 404 的错误页面。可以利用 django.http 下面的 Http404 对象，代码如下：

```
01 from django.shortcuts import render
02 from django.http import HttpResponse, HttpResponseRedirect, Http404
03 from app1.forms import PersonForm
04 from app1.models import Person
05
06 def person_detail(request, pk):        # URL 参数 pk
07     try:
08         p = Person.objects.get(pk=pk)    # 获取 Person 数据
09     except Person.DoesNotExist:
10         raise Http404('Person Does Not Exist')  # 获取不到，抛出 Http404 错误页面
11 return render(request, 'person_detail.html', {'person': p})  # 返回详细信息视图
```

在浏览器中输入 http://127.0.0.1:8000/app1/person_detail/100 会抛出异常，效果如图 14.16 所示。

127.0.0.1:8000/app1/person_detail/100/
Page not found (404)
Request Method: GET
Request URL: http://127.0.0.1:8000/app1/person_detail/100/
Raised by: app1.views.person_detail
Person Does Not Exist
You're seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.

图 14.16　定义 HTTP 404 错误页面

下面讲解一个基于类的视图实例（CBV），基于类的视图非常简单，与基于函数的视图大同小异。首先定义一个类视图，这个类视图需要继承一个基础的类视图，所有的类视图都继承自 views.View。类视图的初始化参数需要给出。将 14.1.7 节定义的 get_name()方法改成基于类的视图，代码如下：

```
01 from django.shortcuts import render
02 from django.http import HttpResponse, HttpResponseRedirect, Http404
03 from django.views import View
04 from app1.forms import PersonForm
05 from app1.models import Person
06
07 class PersonFormView(View):
08     form_class = PersonForm  # 定义表单类
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    initial = {'key': 'value'}      # 定义表单初始化展示参数
    template_name = 'name.html'     # 定义渲染的模板

    def get(self, request, *args, **kwargs):    # 定义 GET 请求的方法
        # 渲染表单
        return render(request, self.template_name,
                      {'form': self.form_class(initial=self.initial)})

    def post(self, request, *args, **kwargs):   # 定义 POST 请求的方法
        form = self.form_class(request.POST)    # 填充表单实例
        if form.is_valid():                     # 判断请求是否有效
            # 使用 form.cleaned_data 获取请求的数据
            first_name = form.cleaned_data['first_name']
            last_name = form.cleaned_data['last_name']
            # 响应拼接后的字符串
            return HttpResponse(first_name + '' + last_name) # 返回拼接的字符串
        # 如果表单无效，返回表单
        return render(request, self.template_name, {'form': form})
```

接下来定义一个 URL，代码如下：

```
from django.urls import path
from app1 import views as app1_views
urlpatterns = [
    path('get_name', app1_views.get_name),
    path('get_name1', app1_views.PersonFormView.as_view()),
    path('person_detail/<int:pk>/', app1_views.person_detail),
]
```

说明：form_class 是指定类使用的表单，template_name 是指定视图渲染的模板。

在浏览器中请求/app1/get_name，会调用 PersonFormViews 视图的方法，如图 14.17 所示。

图 14.17　请求定义的视图

输入 hugo 和 zhang，并单击“提交”按钮，效果如图 14.18 所示。

图 14.18　请求视图结果

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

14.1.9 Django 模板

Django 指定的模板引擎在 settings.py 文件中定义，代码如下：

```
TEMPLATES = [{
       # 模板引擎，默认为 Django 模板
       'BACKEND': 'django.template.backends.django.DjangoTemplates',
       'DIRS': [],          # 模板所在的目录
       'APP_DIRS': True,    # 是否启用 APP 目录
       'OPTIONS': {
       },
    },
]
```

下面通过一个简单的例子，介绍如何使用模板，代码如下：

```
{% extends "base_generic.html" %}
{% block title %}{{ section.title }}{% endblock %}
{% block content %}
<h1>{{ section.title }}</h1>
{% for story in story_list %}
<h2>
  <a href="{{ story.get_absolute_url }}">
    {{ story.headline|upper }}
  </a>
</h2>
<p>{{ story.tease|truncatewords:"100" }}</p>
{% endfor %}
{% endblock %}
```

Django 模板引擎使用“{%%}”来描述，Python 语句区别于<HTML>标签，使用“{{}}”来描述 Python 变量。上面代码中的标签及说明如表 14.5 所示。

表 14.5　Django 模板引擎中的标签说明

标　签	说　明
{% extends ‘base_generic.html’%}	扩展一个母模板
{%block title%}	指定母模板中的一段代码块，此处为 title，在母模板中定义 title 代码块，可以在子模板中重写该代码块。block 标签必须是封闭的，要由{% endblock %}结尾
{{section.title}}	获取变量的值
{% for story in story_list %}、{% endfor %}	和 Python 中的 for 循环用法相似，必须是封闭的

Django 模板的过滤器非常实用，用来将返回的变量值做一些特殊处理，常用的过滤器如下。

- {{value|default:"nothing"}}：用来指定默认值。
- {{value|length}}：用来计算返回的列表或者字符串长度。
- {{value|filesizeformat}}：用来将数字转换成人类可读的文件大小，如 13KB、128MB 等。

電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- {{value|truncatewords:30}}：用来将返回的字符串取固定的长度，此处为 30 个字符。
- {{value|lower}}：用来将返回的数据变为小写字母。

14.2　小结

本章主要介绍 Python 的 Web 开发框架中 Django 框架的使用，首先介绍了该框架的安装，然后又通过创建项目的方式介绍整个框架的使用流程。希望读者可以根据项目的需求，选择本章以及在上一章中学习的 Web 开发框架。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 15 章

Pygame 游戏框架

Python 非常受欢迎的一个原因是它的应用领域非常广泛，其中就包括游戏开发，而使用 Python 进行游戏开发的首选模块就是 Pygame，本章我们就来学习一下如何使用 Pygame 开发游戏。与其他章节不同的是，本章的侧重点不是讲解理论知识，而是在编写游戏的过程中学习 Pygame。

15.1 初识Pygame

Pygame 是跨平台 Python 模块，专为电子游戏设计，包含图像、声音，创建在 SDL（Simple DirectMedia Layer）的基础上，允许实时电子游戏研发而无须被低级语言（如 C 语言或更低级的汇编语言）束缚。基于这样一个设想，所有需要的游戏功能和理念（主要是图像方面）完全简化为游戏逻辑本身，所有的资源结构都可以由高级语言（如 Python）提供。

15.1.1 安装 Pygame

Pygame 的官方网址是 www.pygame.org，在该网址中可以查找 Pygame 相关文档。Pygame 的安装非常简单，只需要如下一行命令即可：

```
pip  install pygame
```

运行结果如图 15.1 所示。

接下来，检测一下 Pygame 是否安装成功。打开 IDLE，输入如下命令：

```
import pygame
pygame.ver
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

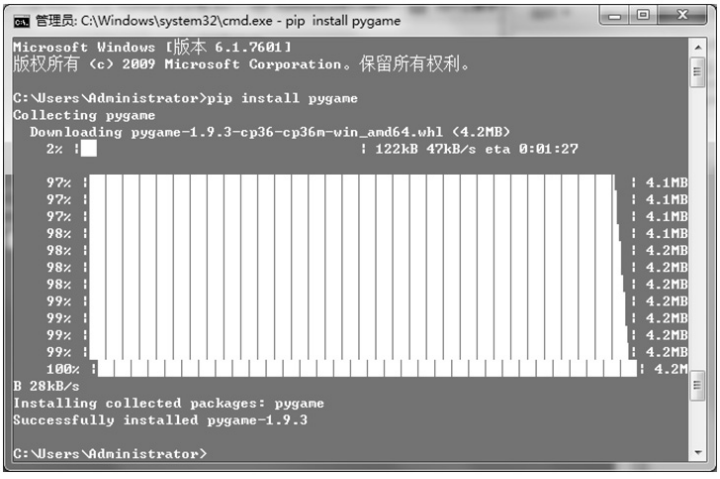


图 15.1　安装 Pygame

如果运行结果如图 15.2 所示，则说明安装成功。

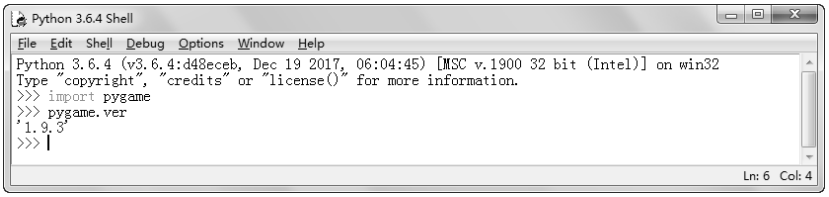

Python 3.6.4 Shell
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import pygame
>>> pygame.ver
'1.9.3'
>>>
Ln: 6 Col: 4

图 15.2　安装成功

15.1.2　Pygame 常用模块

Pygame 做游戏开发的优势在于不需要过多地考虑与底层相关的内容，而可以把工作重心放在游戏逻辑上。例如，Pygame 中集成了很多和底层相关的模块，如访问显示设备、管理事件、使用字体等。Pygame 常用模块如表 15.1 所示。

表 15.1　Pygame 常用模块

模 块 名	功　　能
pygame.cdrom	访问光驱
pygame.cursors	加载光标
pygame.display	访问显示设备
pygame.draw	绘制形状、线和点
pygame.event	管理事件
pygame.font	使用字体
pygame.image	加载和存储图片
pygame.joystick	使用游戏手柄或者类似的东西
pygame.key	读取键盘按键
pygame.mixer	声音
pygame.mouse	鼠标

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

模 块 名	功　能
pygame.movie	播放视频
pygame.music	播放音频
pygame.overlay	访问高级视频叠加
pygame.rect	管理矩形区域
pygame.sndarray	操作声音数据
pygame.sprite	操作移动图像
pygame.surface	管理图像和屏幕
pygame.surfarray	管理点阵图像数据
pygame.time	管理时间和帧信息
pygame.transform	缩放和移动图像

下面，使用 Pygame 的 display 和 event 模块创建一个 Pygame 窗口，代码如下：

```
01 # -*- coding:utf-8 -*-
02 import sys      # 导入 sys 模块
03 import pygame   # 导入 Pygame 模块
04
05 pygame.init()   # 初始化 Pygame
06 size = width, height = 320, 240        # 设置窗口
07 screen = pygame.display.set_mode(size) # 显示窗口
08
09 # 执行死循环，确保窗口一直显示
10 while True:
11     # 检查事件
12     for event in pygame.event.get():    # 遍历所有事件
13         if event.type == pygame.QUIT:   # 如果单击关闭窗口，则退出
14             sys.exit()
15
16 pygame.quit() # 退出 Pygame
```

运行结果如图 15.3 所示。

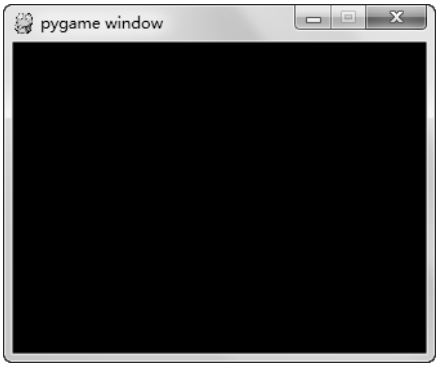


图 15.3　Pygame 创建游戏窗口

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

15.2　Pygame基本使用

Pygame 有很多模块，每个模块又有很多方法，在此不能够逐一讲解，所以，我们通过一个实例来学习 Pygame，然后分解代码，讲解代码中的模块。

创建一个游戏窗口，并在窗口内创建一只小球。以一定的速度移动小球，当小球碰到游戏窗口的边缘时，小球弹回，继续移动。可以按照如下步骤实现该功能。

（1）创建一个游戏窗口，宽和高设置为 640 像素×480 像素，代码如下：

```
01 import sys      # 导入 sys 模块
02 import pygame   # 导入 Pygame 模块
03
04 pygame.init()   # 初始化 Pygame
05 size = width, height = 640, 480        # 设置窗口
06 screen = pygame.display.set_mode(size) # 显示窗口
```

在上述代码中，首先导入 Pygame 模块，然后调用 init()方法初始化 Pygame 模块，接下来设置窗口的宽和高，最后使用 display 模块显示窗口。display 模块的常用方法如表 15.2 所示。

表 15.2　display 模块的常用方法

方 法 名	功　能
pygame.dispaly.init	初始化 display 模块
pygame.dispaly.quit	结束 display 模块
pygame.dispaly.get_init	如果 display 模块已经被初始化，则返回 True
pygame.dispaly.set_mode	初始化一个准备显示的界面
pygame.dispaly.get_surface	获取当前的 Surface 对象
pygame.dispaly.flip	更新整个待显示的 Surface 对象到屏幕上
pygame.dispaly.update	更新部分内容显示到屏幕上。如果没有参数，则与 flip 方法的功能相同

（2）运行上述代码，会出现一个一闪而过的黑色窗口，这是因为程序执行完成后，会自动关闭窗口。如果要让窗口一直显示，则需要使用 while True 让程序一直执行。此外，还需要设置关闭按钮。具体代码如下：

```
01 # -*- coding:utf-8 -*-
02 import sys      # 导入 sys 模块
03 import pygame   # 导入 Pygame 模块
04
05 pygame.init()   # 初始化 Pygame
06 size = width, height = 640, 480        # 设置窗口
07 screen = pygame.display.set_mode(size) # 显示窗口
08
09 # 执行死循环，确保窗口一直显示
10 while True:
11     # 检查事件
12     for event in pygame.event.get():
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
13        if event.type == pygame.QUIT:    # 如果单击关闭窗口，则退出
14            sys.exit()
15
16  pygame.quit()                          # 退出 Pygame
```

在上述代码中，添加了轮询事件检测。pygame.event.get()能够获取事件队列，使用 for…in 遍历事件，然后根据 type 属性判断事件类型。这里的事件处理方式与 GUI 类似，如 event.tpye== pygame.QUIT 表示检测到关闭 Pygame 窗口事件，pygame.KEYDOWN 表示键盘按下事件，pygame.MOUSEBUTTONDOWN 表示鼠标按下事件等。

（3）在窗口中添加小球。我们先准备好一张 ball.png 图片，然后加载该图片，最后将图片显示在窗口中。具体代码如下：

```
01  # -*- coding:utf-8 -*-
02  import sys      # 导入 sys 模块
03  import pygame   # 导入 Pygame 模块
04
05  pygame.init()   # 初始化 Pygame
06  size = width, height = 640, 480        # 设置窗口
07  screen = pygame.display.set_mode(size) # 显示窗口
08  color = (0, 0, 0)                      # 设置颜色
09
10  ball = pygame.image.load("ball.png")   # 加载图片
11  ballrect = ball.get_rect()             # 获取矩形区域
12
13  # 执行死循环，确保窗口一直显示
14  while True:
15      # 检查事件
16      for event in pygame.event.get():
17          if event.type == pygame.QUIT:    # 如果单击关闭窗口，则退出
18              sys.exit()
19
20      screen.fill(color)                   # 填充颜色
21      screen.blit(ball, ballrect)          # 将图片画到窗口上
22      pygame.display.flip()                # 更新全部显示
23
24  pygame.quit()                            # 退出 Pygame
```

在上述代码中，使用 image 模块的 load()方法加载图片，返回值 ball 是一个 Surface 对象。Surface 是用来代表图片的 Pygame 对象，可以对一个 Surface 对象进行涂画、变形、复制等各种操作。事实上，屏幕也只是一个 Surface 对象，pygame.display.set_mode()方法就返回了一个屏幕 Surface 对象。如果将 ball 这个 Surface 对象画到 screen Surface 对象中，则需要使用 blit()方法，然后使用 display 模块的 flip()方法更新整个待显示的 Surface 对象到屏幕上。Surface 对象的常用方法如表 15.3 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

表 15.3　Surface 对象的常用方法

方 法 名	功　　能
pygame.Surface.blit	将一幅图像画到另一幅图像上
pygame.Surface.convert	转换图像的像素格式
pygame.Surface.convert_alpha	转换图像的像素格式，包含 Alpha 通道的转换
pygame.Surface.fill	使用颜色填充 Surface
pygame.Surface.get_rect	获取 Surface 的矩形区域

运行上述代码，结果如图 15.4 所示。

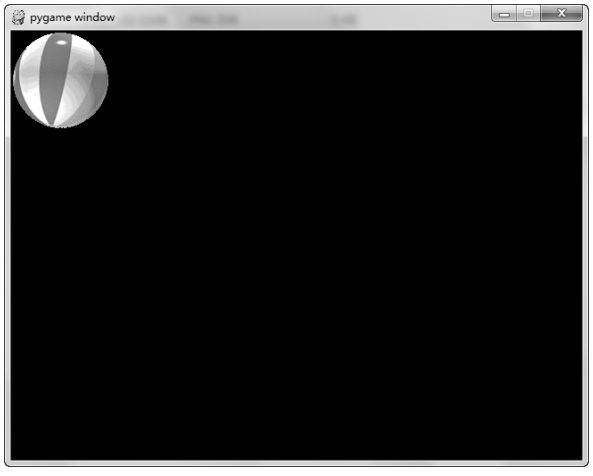


图 15.4　在窗口中添加小球

（4）下面该让小球动起来了。ball.get_rect()方法的返回值 ballrect 是一个 Rect 对象，该对象有一个 move()方法可以用于移动矩形。move(x,y)函数有两个参数，第一个参数是 *X* 轴移动的距离，第二个参数是 *Y* 轴移动的距离。窗口左上角坐标为(0,0)，如果 move(100,50)，则移动后的坐标位置如图 15.5 所示。

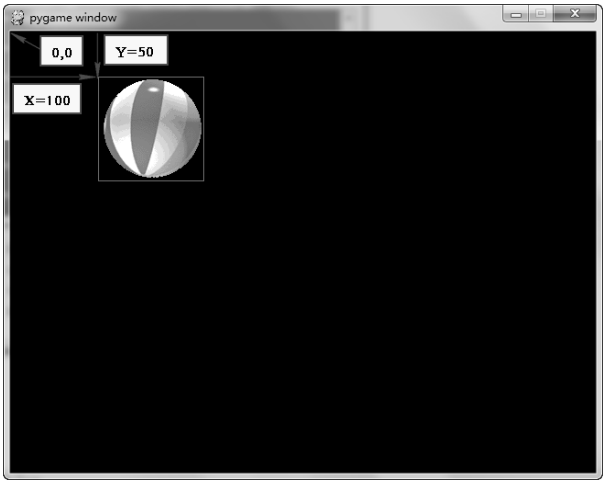


图 15.5　移动后的坐标位置

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

为实现小球不停地移动，将 move()方法添加到 while 循环内。具体代码如下：

```
01 # -*- coding:utf-8 -*-
02 import sys                                # 导入 sys 模块
03 import pygame                             # 导入 Pygame 模块
04
05 pygame.init()                            # 初始化 Pygame
06 size = width, height = 640, 480          # 设置窗口
07 screen = pygame.display.set_mode(size) # 显示窗口
08 color = (0, 0, 0)                        # 设置颜色
09
10 ball = pygame.image.load("ball.png")    # 加载图片
11 ballrect = ball.get_rect()               # 获取矩形区域
12
13 speed = [5,5]                            # 设置移动的 X 轴、Y 轴距离
14 # 执行死循环，确保窗口一直显示
15 while True:
16     # 检查事件
17     for event in pygame.event.get():
18         if event.type == pygame.QUIT:    # 如果单击关闭窗口，则退出
19             sys.exit()
20
21     ballrect = ballrect.move(speed)     # 移动小球
22     screen.fill(color)                  # 填充颜色
23     screen.blit(ball, ballrect)         # 将图片画到窗口上
24     pygame.display.flip()               # 更新全部显示
25
26 pygame.quit()                           # 退出 Pygame
```

（5）运行上述代码，发现小球在屏幕中一闪而过。小球并没有真正消失，而是移动到窗口之外，此时需要添加碰撞检测功能。如果小球与窗口的任一边缘发生碰撞，则更改小球的移动方向。具体代码如下：

```
01 # -*- coding:utf-8 -*-
02 import sys                                # 导入 sys 模块
03 import pygame                             # 导入 Pygame 模块
04
05 pygame.init()                            # 初始化 Pygame
06 size = width, height = 640, 480          # 设置窗口
07 screen = pygame.display.set_mode(size) # 显示窗口
08 color = (0, 0, 0)                        # 设置颜色
09
10 ball = pygame.image.load("ball.png")    # 加载图片
11 ballrect = ball.get_rect()               # 获取矩形区域
12
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
speed = [5,5]                                # 设置移动的 X 轴、Y 轴距离
# 执行死循环，确保窗口一直显示
while True:
    # 检查事件
    for event in pygame.event.get():
        if event.type == pygame.QUIT:        # 如果单击关闭窗口，则退出
            sys.exit()

    ballrect = ballrect.move(speed)          # 移动小球
    # 碰到左右边缘
    if ballrect.left < 0 or ballrect.right > width:
        speed[0] = -speed[0]
    # 碰到上下边缘
    if ballrect.top < 0 or ballrect.bottom > height:
        speed[1] = -speed[1]

    screen.fill(color)                       # 填充颜色
    screen.blit(ball, ballrect)              # 将图片画到窗口上
    pygame.display.flip()                    # 更新全部显示

pygame.quit()                                # 退出 Pygame
```

在上述代码中，添加了碰撞检测功能。如果小球碰到左右边缘，则更改 *X* 轴数据为负数；如果小球碰到上下边缘，则更改 *Y* 轴数据为负数。运行结果如图 15.6 所示。

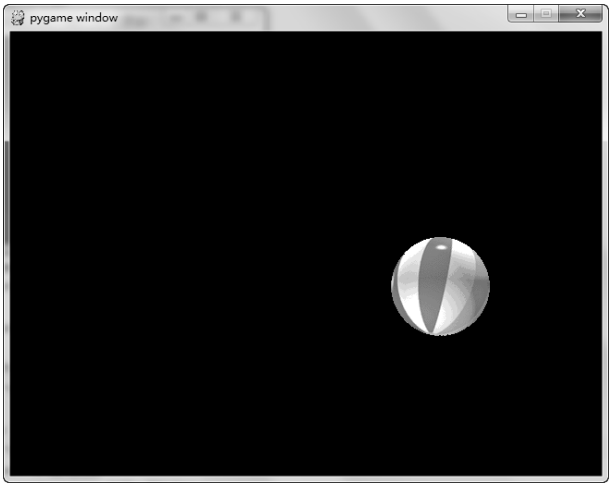


图 15.6　小球不停地跳跃

（6）运行上述代码，发现好像有多只小球在飞快地移动，这是因为运行上述代码的时间非常短，导致肉眼观察出现错觉，因此需要添加一个“时钟”来控制程序运行的时间。这时就需要使用 Pygame 的 time 模块。在使用 Pygame 时钟之前，必须先创建 Clock 对象的一个实例，然后在 while 循环中设置多长时间运行一次。具体代码如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
# -*- coding:utf-8 -*-
import sys                              # 导入 sys 模块
import pygame                           # 导入 Pygame 模块

pygame.init()                           # 初始化 Pygame
size = width, height = 640, 480         # 设置窗口
screen = pygame.display.set_mode(size) # 显示窗口
color = (0, 0, 0)                       # 设置颜色

ball = pygame.image.load("ball.png")    # 加载图片
ballrect = ball.get_rect()              # 获取矩形区域

speed = [5,5]                           # 设置移动的 X 轴、Y 轴距离
clock = pygame.time.Clock()             # 设置时钟
# 执行死循环，确保窗口一直显示
while True:
    clock.tick(60)                      # 每秒执行 60 次
    # 检查事件
    for event in pygame.event.get():
        if event.type == pygame.QUIT:   # 如果单击关闭窗口，则退出
            sys.exit()

    ballrect = ballrect.move(speed)     # 移动小球
    # 碰到左右边缘
    if ballrect.left < 0 or ballrect.right > width:
        speed[0] = -speed[0]
    # 碰到上下边缘
    if ballrect.top < 0 or ballrect.bottom > height:
        speed[1] = -speed[1]

    screen.fill(color)                  # 填充颜色
    screen.blit(ball, ballrect)         # 将图片画到窗口上
    pygame.display.flip()               # 更新全部显示

pygame.quit()                           # 退出 Pygame
```

至此，就完成了跳跃的小球游戏。

15.3 小结

本章主要讲解如何使用 Pygame 开发游戏，其中通过一个跳跃的小球游戏来了解 Pygame 的基本使用。通过对本章的学习，希望读者可以掌握 Pygame 的基础知识，并使用 Python 面向对象的思维方式开发一个 Python 小游戏，进一步体会 Python 编程的乐趣。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 16 章

网络爬虫框架

随着大数据时代的来临，网络信息量也变得更多，网络爬虫在互联网中的地位将越来越重要。本章将介绍通过 Python 语言实现网络爬虫的常用技术，以及常见的网络爬虫框架。

16.1 初识网络爬虫

16.1.1 网络爬虫概述

网络爬虫（又被称为网络蜘蛛、网络机器人，在某社区中经常被称为网页追逐者）可以按照指定的规则（网络爬虫的算法）自动浏览或抓取互联网中的信息。通过 Python 可以很轻松地编写爬虫程序或者脚本。

在生活中，网络爬虫经常出现，搜索引擎就离不开网络爬虫。例如，百度搜索引擎的爬虫名字叫作百度蜘蛛（Baidu Spider）。百度蜘蛛是百度搜索引擎的一个自动程序，它每天都会在海量的互联网信息中进行爬取，收集并整理互联网上的网页、图片、视频等信息。当用户在百度搜索引擎中输入对应的关键词时，百度将从收集的互联网信息中找出相关的内容，按照一定的顺序将信息展现给用户。百度蜘蛛在工作的过程当中，搜索引擎会构建一个调度程序来调度百度蜘蛛的工作，这些调度程序都是需要使用一定的算法来实现的。采用不同的算法，爬虫的工作效率会有所不同，爬取的结果也会有所差异。所以，在学习爬虫的时候，不仅需要了解爬虫的实现过程，还需要了解一些常见的爬虫算法。在特定的情况下，还需要开发者自己编写相应的算法。

16.1.2 网络爬虫的分类

网络爬虫按照实现的技术和结构可以分为通用网络爬虫、聚焦网络爬虫、增量式网络爬虫、深层网络爬虫等类型。在实际的网络爬虫中，通常是这几类爬虫的组合体。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

1．通用网络爬虫

通用网络爬虫又叫作全网爬虫（Scalable Web Crawler），它的爬行范围和爬取的数据量巨大。正是由于其爬取的数据是海量数据，所以对于爬行速度和存储空间要求较高。通用网络爬虫在爬行页面的顺序上要求相对较低，同时由于待刷新的页面太多，通常采用并行工作方式，所以需要较长时间才可以刷新一次页面。这种网络爬虫主要应用于大型搜索引擎中，有非常高的应用价值。通用网络爬虫主要由初始 URL 集合、URL 队列、页面爬行模块、页面分析模块、页面数据库、链接过滤模块等构成。

2．聚焦网络爬虫

聚焦网络爬虫（Focused Crawler）也叫主题网络爬虫（Topical Crawler），是指按照预先定义的主题，有选择地进行相关网页爬取的一种爬虫。它和通用网络爬虫相比，不会将目标资源定位在整个互联网当中，而是将爬取的目标网页定位在与主题相关的页面中，极大地节省了硬件和网络资源，保存页面的速度也更快了。聚焦网络爬虫主要应用于对特定信息的爬取，为某类特定的人群提供服务。

3．增量式网络爬虫

所谓增量式，对应着增量式更新，增量式更新指的是在更新的时候只更新改变的地方，未改变的地方则不更新，所以增量式网络爬虫（Incremental Web Crawler）在爬取网页的时候，只会在需要的时候爬行新产生或发生变化的页面，对于没有发生变化的页面则不会爬取。这样可以有效减少数据下载量，降低时间和空间上的耗费，但是在爬行算法上增加了一些难度。

4．深层网络爬虫

在互联网中，Web 页面按存在方式可以分为表层网页（Surface Web）和深层网页（Deep Web）。表层网页指的是不需要提交表单，使用静态的超链接就可以直接访问的静态页面。深层网页指的是那些大部分内容不能通过静态链接获取的、隐藏在搜索表单后面的、需要用户提交一些关键词才能获取的 Web 页面。深层网页需要访问的信息数量是表层网页需要访问的信息数量的几百倍，所以深层网页是主要的爬取对象。

深层网络爬虫主要由 6 个基本功能模块（爬行控制器、解析器、表单分析器、表单处理器、响应分析器、LVS 控制器）和两个爬虫内部数据结构（URL 列表、LVS 表）等部分构成。其中，LVS（Label Value Set）表示标签/数值集合，用来表示填充表单的数据源。

16.1.3　网络爬虫的基本原理

一个通用的网络爬虫基本工作流程如图 16.1 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

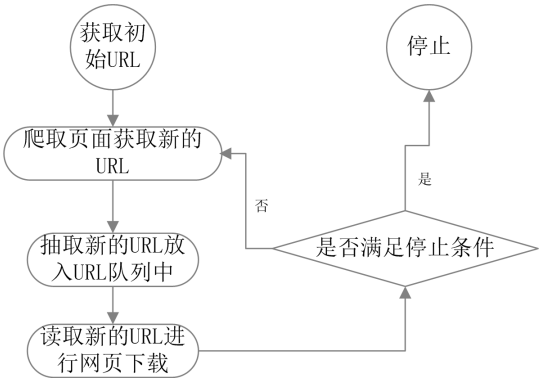


图 16.1　通用的网络爬虫基本工作流程

网络爬虫的基本工作流程如下。

（1）获取初始 URL，该 URL 是用户自己指定的初始爬取的网页。

（2）在爬取对应 URL 的网页时，获取新的 URL。

（3）将新的 URL 放入 URL 队列中。

（4）从 URL 队列中读取新的 URL，然后依据新的 URL 爬取网页，同时从新的网页中获取新的 URL，重复上述爬取过程。

（5）设置停止条件。如果没有设置停止条件，那么爬虫会一直爬取下去，直到无法获取新的 URL 为止。当设置了停止条件后，爬虫将会在满足停止条件时停止爬取。

16.2　网络爬虫的常用技术

16.2.1　Python 的网络请求

在上一节中多次提到了 URL 与下载网页，这两项是网络爬虫必备而又关键的功能。说到这两个功能，必然离不开与 HTTP 打交道。本节将介绍在 Python 中实现 HTTP 网络请求常见的 3 种方式：urllib、urllib3 及 requests。

1．urllib 模块

urllib 是 Python 自带模块，该模块中提供了一个 urlopen()方法，通过该方法指定 URL 发送网络请求来获取数据。urllib 提供了多个子模块，具体的模块名称及描述如表 16.1 所示。

表 16.1　urllib 中的子模块

模块名称	描　　述
urllib.request	该模块定义了打开 URL（主要是 HTTP）的方法和类，例如，身份验证、重定向、Cookie 等
urllib.error	该模块中主要包含异常类，基本的异常类是 URLError
urllib.parse	该模块定义的功能分为两大类：URL 解析和 URL 引用
urllib.robotparser	该模块用于解析 robots.txt 文件

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

通过 urllib.request 模块实现发送请求并读取网页内容的简单示例如下：

```
01 import urllib.request       # 导入模块
02
03 # 打开指定需要爬取的网页
04 response = urllib.request.urlopen('http://www.baidu.com')
05 html = response.read()      # 读取网页代码
06 print(html)                 # 打印读取内容
```

在上面的示例中，是通过 GET 请求方式获取百度的网页内容的。下面通过使用 urllib.request 模块的 POST 请求实现获取网页信息的内容，示例如下：

```
01 import urllib.parse
02 import urllib.request
03
04 # 将数据使用 urlencode 编码处理后，再使用 encoding 设置为 UTF-8 编码
05 data = bytes(urllib.parse.urlencode({'word': 'hello'}), encoding='utf8')
06 # 打开指定需要爬取的网页
07 response = urllib.request.urlopen('http://httpbin.org/post', data=data)
08 html = response.read()      # 读取网页代码
09 print(html)                 # 打印读取内容
```

> **说明：**这里通过 http://httpbin.org/post 网站进行演示，该网站可以作为练习使用 urllib 的一个站点来模拟各种请求操作。

2．urllib3 模块

urllib3 是一个功能强大、条理清晰、用于 HTTP 客户端的 Python 库，许多 Python 的原生系统已经开始使用 urllib3。urllib3 提供了很多 Python 标准库里所没有的重要特性，例如：

- 线程安全。
- 连接池。
- 客户端 SSL / TLS 验证。
- 使用多部分编码上传文件。
- Helpers 用于重试请求并处理 HTTP 重定向。
- 支持 gzip 和 deflate 编码。
- 支持 HTTP 和 SOCKS 代理。
- 100%的测试覆盖率。

通过 urllib3 模块实现发送网络请求的示例代码如下：

```
01 import urllib3
02
03 # 创建 PoolManager 对象，用于处理与线程池的连接及线程安全的所有细节
04 http = urllib3.PoolManager()
05 # 对需要爬取的网页发送请求
06 response = http.request('GET','https://www.baidu.com/')
07 print(response.data)         # 打印读取内容
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

以 POST 请求方式实现获取网页信息的内容，关键代码如下：

```
# 对需要爬取的网页发送请求
response = http.request('POST',
                    'http://httpbin.org/post'
                    ,fields={'word': 'hello'})
```

> **注意：**在使用 urllib3 模块前，需要在 Python 中通过执行 pip install urllib3 代码进行该模块的安装。

3．requests 模块

requests 是 Python 中实现 HTTP 请求的一种方式。requests 是第三方模块，该模块在实现 HTTP 请求时要比 urllib 模块简化很多，操作更加人性化。在使用 requests 模块前，需要通过执行 pip install requests 代码进行该模块的安装。requests 模块的功能特性如下：

- Keep-Alive &连接池。
- 国际化域名和 URL。
- 带持久 Cookie 的会话。
- 浏览器式的 SSL 认证。
- 自动内容解码。
- 基本/摘要式的身份认证。
- 优雅的 key/value Cookie。
- 自动解压。
- Unicode 响应体。
- HTTP(S)代理支持。
- 文件分块上传。
- 流下载。
- 连接超时。
- 分块请求。
- 支持.netrc。

以 GET 请求方式打印多种请求信息的示例代码如下：

```
import requests                 # 导入模块

response = requests.get('http://www.baidu.com')
print(response.status_code)     # 打印状态码
print(response.url)             # 打印请求 URL
print(response.headers)         # 打印头部信息
print(response.cookies)         # 打印 Cookie 信息
print(response.text)            # 以文本形式打印网页源码
print(response.content)         # 以字节流形式打印网页源码
```

以 POST 请求方式发送 HTTP 网络请求的示例代码如下：

```
import requests

data = {'word': 'hello'}        # 表单参数
# 对需要爬取的网页发送请求
response = requests.post('http://httpbin.org/post', data=data)
print(response.content)         # 以字节流形式打印网页源码
```

requests 模块不仅提供了以上两种常用的请求方式，还提供了以下多种网络请求方式。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
requests.put('http://httpbin.org/put',data = {'key':'value'}) # PUT 请求
requests.delete('http://httpbin.org/delete')                   # DELETE 请求
requests.head('http://httpbin.org/get')                        # HEAD 请求
requests.options('http://httpbin.org/get')                     # OPTIONS 请求
```

如果发现请求的 URL 地址中参数是跟在?（问号）的后面的，例如，httpbin.org/get?key=val，那么 requests 模块允许使用 params 关键字参数以一个字符串字典来提供这些参数。例如，要想传递 key1=value1 和 key2=value2 到 httpbin.org/get 中，则可以使用如下代码：

```
import requests

payload = {'key1': 'value1', 'key2': 'value2'}     # 传递的参数
# 对需要爬取的网页发送请求
response = requests.get("http://httpbin.org/get", params=payload)
print(response.content)                             # 以字节流形式打印网页源码
```

16.2.2 请求头部处理

在请求一个网页的内容时，有时会发现，无论通过 GET、POST 或其他请求方式，都会出现 403 错误。这是因为服务器拒绝了你的访问，这是网页为了防止恶意采集信息所使用的反爬虫设置。此时可以通过模拟浏览器的头部信息来进行访问，就能解决以上反爬虫设置的问题。下面以 requests 模块为例，介绍请求头部处理，具体步骤如下。

（1）通过浏览器的网络监视器查看头部信息。首先通过火狐浏览器打开对应的网页地址，然后按快捷键<Ctrl+Shift+E>打开网络监视器，接着刷新当前页面，网络监视器中将显示如图 16.2 所示的数据变化。

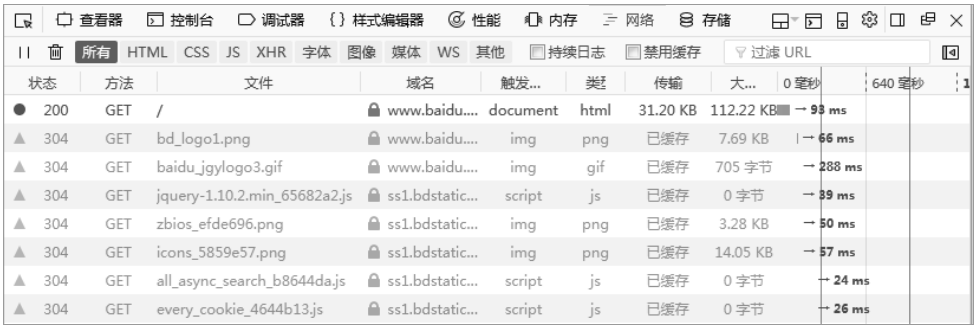

查看器 控制台 调试器 样式编辑器 性能 内存 网络 存储
所有 HTML CSS JS XHR 字体 图像 媒体 WS 其他 持续日志 禁用缓存 过滤 URL
状态 方法 文件 域名 触发... 类型 传输 大... 0 毫秒 640 毫秒
200 GET / www.baidu.... document html 31.20 KB 112.22 KB → 93 ms
304 GET bd_logo1.png www.baidu.... img png 已缓存 7.69 KB → 66 ms
304 GET baidu_jgylogo3.gif www.baidu.... img gif 已缓存 705 字节 → 288 ms
304 GET jquery-1.10.2.min_65682a2.js ss1.bdstatic... script js 已缓存 0 字节 → 89 ms
304 GET zbios_efde696.png ss1.bdstatic... img png 已缓存 3.28 KB → 50 ms
304 GET icons_5859e57.png ss1.bdstatic... img png 已缓存 14.05 KB → 57 ms
304 GET all_async_search_b8644da.js ss1.bdstatic... script js 已缓存 0 字节 → 24 ms
304 GET every_cookie_4644b13.js ss1.bdstatic... script js 已缓存 0 字节 → 26 ms

图 16.2　网络监视器中显示的数据变化

（2）选中第一条信息，在右侧的“消息头”面板中将显示请求头部信息，复制该信息，如图 16.3 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 16.3　复制头部信息

（3）首先创建一个需要爬取的 URL 地址，然后创建头部信息，接着发送网络请求等待响应，最后打印网页源码信息。实现代码如下：

```
01 import requests
02 url = 'https://www.baidu.com/'                          # 创建一个需要爬取的 URL 地址
03 # 创建头部信息
04 headers = {'User-Agent':'OW64; rv:59.0) Gecko/20100101 Firefox/59.0'}
05 response  = requests.get(url, headers=headers)          # 发送网络请求
06 print(response.content)                                 # 以字节流形式打印网页源码
```

16.2.3　网络超时

在访问一个网页时，如果该网页长时间未响应，系统就会判断网络超时，因而无法打开该网页。下面通过代码来模拟一个网络超时的现象。

```
01 import requests
02 # 循环发送请求 50 次
03 for a in range(1, 50):
04     try:    # 捕获异常
05         # 设置超时时间为 0.5 秒
06         response = requests.get('https://www.baidu.com/', timeout=0.5)
07         print(response.status_code)                     # 打印状态码
08     except Exception as e:                              # 捕获异常
09         print('异常'+str(e))                            # 打印异常信息
```

打印结果如图 16.4 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
200
200
200
异常HTTPSConnectionPool(host='www.baidu.com', port=443): Read timed out. (read timeout=1)
200
200
200
```

图 16.4　异常信息

> **说明：**在上面的代码中，模拟了循环发送 50 次请求，并且设置了超时时间为 0.5 秒，所以在 0.5 秒内服务器未做出响应将被视为超时，并将超时信息打印在控制台中。根据以上模拟测试结果，可以确认在不同的情况下应设置不同的 timeout 值。

说起网络异常信息，requests 模块同样提供了 3 种常见的网络异常类，示例代码如下：

```
import requests
# 导入 requests.exceptions 模块中的 3 种网络异常类
from requests.exceptions import ReadTimeout,HTTPError,RequestException
# 循环发送请求 50 次
for a in range(1, 50):
    try:    # 捕获异常
        # 设置超时时间为 0.5 秒
        response = requests.get('https://www.baidu.com/', timeout=0.5)
        print(response.status_code)                    # 打印状态码
    except ReadTimeout:                                # 超时异常
        print('timeout')
    except HTTPError:                                  # HTTP 异常
        print('httperror')
    except RequestException:                           # 请求异常
        print('reqerror')
```

16.2.4　代理服务

在爬取网页的过程中，经常会出现不久前可以爬取的网页现在无法爬取的问题，这是因为你的 IP 被爬取网页的服务器所屏蔽了，此时代理服务可以解决这一麻烦。在设置代理服务时，首先需要找到代理地址，例如，122.114.31.177，对应的端口号为 808，完整的格式为 122.114.31.177:808。示例代码如下：

```
import requests

proxy = {'http': '122.114.31.177:808',
         'https': '122.114.31.177:8080'}  # 设置代理地址与对应的端口号
# 对需要爬取的网页发送请求
response = requests.get('http://www.mingrisoft.com/', proxies=proxy)
print(response.content)  # 以字节流形式打印网页源码
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **注意：**由于示例中的代理地址是免费的，所以使用的时间不固定。如果超出使用的时间范围，则该地址将失效。在代理地址失效或错误时，控制台上将显示如图 16.5 所示的错误信息。

```
Traceback (most recent call last):
  File "C:\Users\Administrator\AppData\Local\Programs\Python\Python36\lib\site-packages\urllib3\connection.py", line 141, in _new_conn
    (self.host, self.port), self.timeout, **extra_kw)
  File "C:\Users\Administrator\AppData\Local\Programs\Python\Python36\lib\site-packages\urllib3\util\connection.py", line 83, in create_connection
    raise err
  File "C:\Users\Administrator\AppData\Local\Programs\Python\Python36\lib\site-packages\urllib3\util\connection.py", line 73, in create_connection
    sock.connect(sa)
TimeoutError: [WinError 10060] 由于连接方在一段时间后没有正确答复或连接的主机没有反应，连接尝试失败。
```

图 16.5　代理地址失效或错误所提示的信息

16.2.5　HTML 解析之 BeautifulSoup

BeautifulSoup 是一个用于从 HTML 和 XML 文件中提取数据的 Python 库。BeautifulSoup 模块中的查找提取功能非常强大，而且非常便捷，通常可以节省程序员数小时甚至数天的工作时间。

BeautifulSoup 自动将输入文档转换为 Unicode 编码，将输出文档转换为 UTF-8 编码。通常不需要考虑编码方式，除非文档没有指定一种编码方式，这时，BeautifulSoup 就不能自动识别编码方式了。这时，仅仅需要说明一下原始编码方式就可以了。

1．BeautifulSoup 的安装

BeautifulSoup 3 已经停止开发，目前推荐使用的是 BeautifulSoup 4，不过它已经被移植到 bs4 当中了，所以在导入时需要先导入 bs4，再导入 BeautifulSoup。安装 BeautifulSoup 有以下 3 种方式。

- 如果你使用的是最新版本的 Debian 或 Ubuntu Linux，则可以使用系统软件包管理器安装 BeautifulSoup，安装命令为 apt-get install python-bs4。
- BeautifulSoup 4 是通过 PyPi 发布的，可以通过 easy_install 或 pip 来安装它。包名是 beautifulsoup4，它可以兼容 Python 2 和 Python 3。安装命令为 easy_install beautifulsoup4 或者 pip install beautifulsoup4。

> **注意：**在使用 BeautifulSoup 4 之前，需要先通过命令 pip install bs4 进行 bs4 库的安装。

- 如果当前的 BeautifulSoup 不是你想要的版本，则可以通过下载源码的方式进行安装。源码的下载地址为 https://www.crummy.com/software/BeautifulSoup/bs4/download。在控制台中打开源码的指定路径，输入命令 python setup.py install 即可，如图 16.6 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

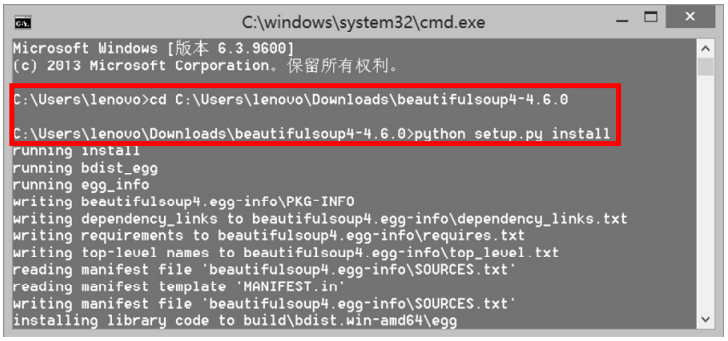


图 16.6　通过源码安装 BeautifulSoup

BeautifulSoup 不仅支持 Python 标准库中包含的 HTML 解析器，而且还支持许多第三方 Python 解析器，其中包含 lxml 解析器。根据不同的操作系统，可以使用以下命令之一安装 lxml 解析器：

- apt-get install python-lxml。
- easy_install lxml。
- pip install lxml。

另一个解析器是 html5lib，它是一个用于解析 HTML 的 Python 库，按照 Web 浏览器的方式解析 HTML。可以使用以下命令之一安装 html5lib 解析器：

- apt-get install python-html5lib。
- easy_install html5lib。
- pip install html5lib。

在表 16.2 中比较了每个解析器的优缺点。

表 16.2　解析器的优缺点比较

解 析 器	用　　法	优　　点	缺　　点
Python 标准库	BeautifulSoup(markup, "html.parser")	Python 标准库 执行速度适中	在 Python 2.7.3 或 3.2.2 之前的版本中，文档容错能力差
lxml 的 HTML 解析器	BeautifulSoup(markup, "lxml")	速度快 文档容错能力强	需要安装 C 语言库
lxml 的 XML 解析器	BeautifulSoup(markup, "lxml-xml") BeautifulSoup(markup, "xml")	速度快 唯一支持 XML 的解析器	需要安装 C 语言库
html5lib	BeautifulSoup(markup, "html5lib")	最好的容错性 以浏览器的方式解析文档 生成 HTML5 格式的文档	速度慢，不依赖外部扩展

2．BeautifulSoup 的使用

BeautifulSoup 安装完成以后，下面将介绍如何通过 BeautifulSoup 库完成 HTML 的解析工作，具体示例步骤如下。

（1）导入 bs4 库，然后创建一个模拟 HTML 代码的字符串，代码如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
01 from bs4 import BeautifulSoup  # 导入 BeautifulSoup 库
02
03 # 创建模拟 HTML 代码的字符串
04 html_doc = """
05 <html><head><title>The Dormouse's story</title></head>
06 <body>
07 <p class="title"><b>The Dormouse's story</b></p>
08
09 <p class="story">Once upon a time there were three little sisters; and their
names were
10 <a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
11 <a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
12 <a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
13 and they lived at the bottom of a well.</p>
14
15 <p class="story">...</p>
16 """
```

（2）创建 BeautifulSoup 对象，并指定解析器为 lxml，然后通过打印的方式将解析的 HTML 代码显示在控制台中。代码如下：

```
01 # 创建一个 BeautifulSoup 对象，获取页面正文
02 soup = BeautifulSoup(html_doc, features="lxml")
03 print(soup)                # 打印解析的 HTML 代码
```

运行结果如图 16.7 所示。

```
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>
<p class="story">Once upon a time there were three little sisters; and their names were
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>
<p class="story">...</p>
</body></html>
```

图 16.7　显示解析后的 HTML 代码

说明：如果将 html_doc 字符串中的代码保存在 index.html 文件中，则可以通过打开 HTML 文件的方式进行代码的解析，并且可以通过 prettify()方法进行代码的格式化处理。代码如下：

```
01   # 创建 BeautifulSoup 对象，打开需要解析的 HTML 文件
02   soup = BeautifulSoup(open('index.html'),'lxml')
03   print(soup.prettify())    # 打印格式化后的代码
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

16.3 网络爬虫开发常用框架

爬虫框架就是一些爬虫项目的半成品，可以先将一些爬虫常用的功能写好，然后留下一些接口，在不同的爬虫项目当中调用适合自己项目的接口，再编写少量的代码实现自己需要的功能。因为框架中已经实现了爬虫常用的功能，所以为开发人员节省了很多精力与时间。

16.3.1 Scrapy 爬虫框架

Scrapy 是一套比较成熟的 Python 爬虫框架，简单轻巧，使用方便，可以高效率地爬取 Web 页面中的信息，并从页面中提取结构化的数据。Scrapy 是一套开源的框架，所以在使用时不需要担心收取费用的问题。Scrapy 的官网地址为 https://scrapy.org，官网页面如图 16.8 所示。

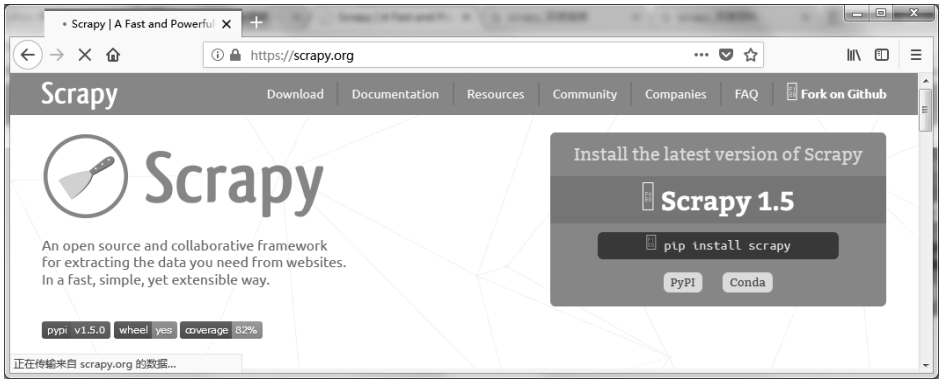


图 16.8 Scrapy 的官网页面

> **说明：**Scrapy 开源框架为开发者提供了非常贴心的开发文档，其中详细介绍了该框架的安装及使用方法。

16.3.2 Crawley 爬虫框架

Crawley 也是 Python 开发出来的爬虫框架，该框架致力于改变人们从互联网中提取数据的方式。Crawley 框架的具体特性如下。

- 它是基于 Eventlet 构建的高速网络爬虫框架。
- 可以将数据存储在关系型数据库中，例如，PostgreSQL、MySQL、Oracle、SQLite。
- 可以将爬取的数据导入为 JSON、XML 格式。
- 支持非关系型数据库，例如，MongoDB 和 CouchDB。
- 支持命令行工具。
- 可以使用你喜欢的工具进行数据的提取，例如，XPath 或 PyQuery 工具。
- 支持使用 Cookie 登录或访问那些只有登录才可以访问的网页。
- 简单易学（可以参照示例）。

Crawley 的官网地址为 http://project.crawley-cloud.com，官网页面如图 16.9 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 16.9　Crawley 的官网页面

16.3.3　PySpider 爬虫框架

相对于 Scrapy 框架而言，PySpider 框架是一支新秀。它采用 Python 语言编写，支持分布式架构，支持多种数据库后端，强大的 WebUI 支持脚本编辑器、任务监视器、项目管理器及结果查看器。PySpider 框架的具体特性如下。

- 采用 Python 脚本控制，可以使用任何你喜欢的 HTML 解析包（内置 PyQuery）。
- 可以在 Web 界面上编写调试脚本、起停脚本、监控执行状态、查看活动历史、获取结果。
- 支持 MySQL、MongoDB、Redis、SQLite、Elasticsearch、PostgreSQL 与 SQLAlchemy。
- 支持 RabbitMQ、Beanstalk、Redis 和 Kombu 作为消息队列。
- 支持抓取 JavaScript 的页面。
- 强大的调度控制，支持超时重爬及优先级设置。
- 组件可替换，支持单机/分布式部署，支持 Docker 部署。

PySpider 的源码地址为 https://github.com/binux/pyspider/releases，开发文档地址为 http://docs.pyspider.org/。

16.4　Scrapy爬虫框架的使用

16.4.1　搭建 Scrapy 爬虫框架

Scrapy 爬虫框架依赖的库比较多，尤其是在 Windows 系统下，至少需要依赖的库有 Twisted、lxml、pyOpenSSL 及 pywin32。搭建 Scrapy 爬虫框架的具体步骤如下。

1．安装 Twisted 模块

（1）打开 Python 扩展包的非官方 Windows 二进制文件网站（https://www.lfd.uci.edu/~gohlke/pythonlibs），然后使用快捷键<Ctrl+F>搜索“twisted”模块，单击对应的索引，如图 16.10 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

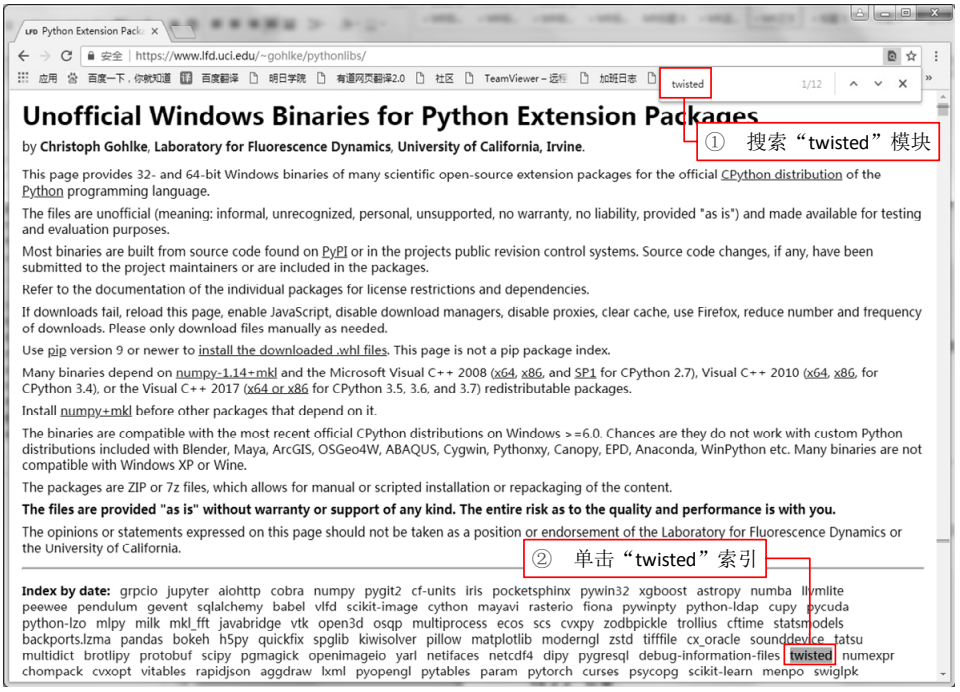


图 16.10 单击“twisted”索引

（2）单击“twisted”索引以后，网页将自动定位到下载“twisted”扩展包二进制文件的位置，然后根据自己的 Python 版本进行下载即可。由于笔者使用的是 Python 3.6，所以这里单击“Twisted-18.4.0-cp36-cp36m-win_amd64.whl”进行下载，其中“cp36”代表对应 Python 3.6 版本，“win32”与“win_amd64”分别表示 Windows 32 位与 64 位系统，如图 16.11 所示。

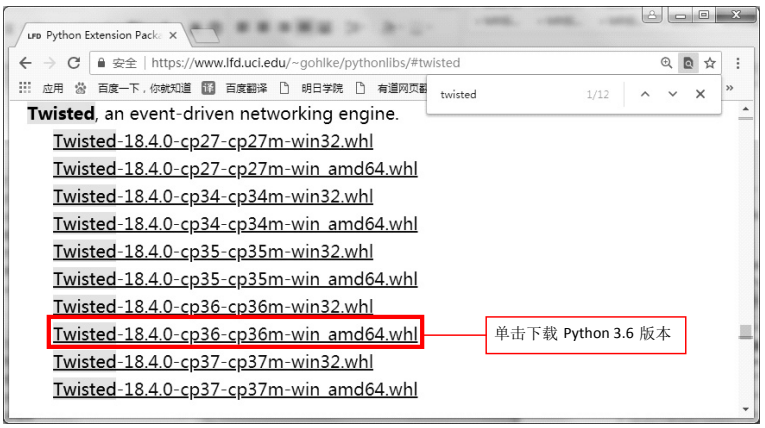


图 16.11 下载“Twisted-18.4.0-cp36-cp36m-win_amd64.whl”二进制文件

（3）“Twisted-18.4.0-cp36-cp36m-win_amd64.whl”二进制文件下载完成后，在下载文件的路径下，在按下<Shift>键的同时单击鼠标右键，在弹出的快捷菜单中选择“在此处打开命令窗口”命令，如图 16.12 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

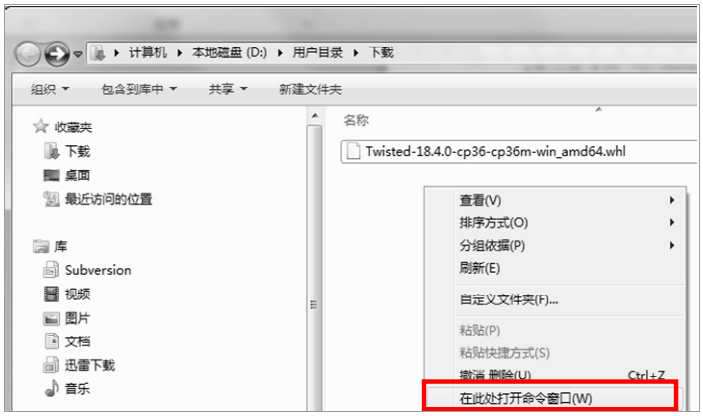


图 16.12　在下载文件的路径下打开命令行窗口

（4）在打开的命令行窗口中输入“pip install Twisted-18.4.0-cp36-cp36m-win_amd64.whl”命令，安装 Twisted 模块，如图 16.13 所示。

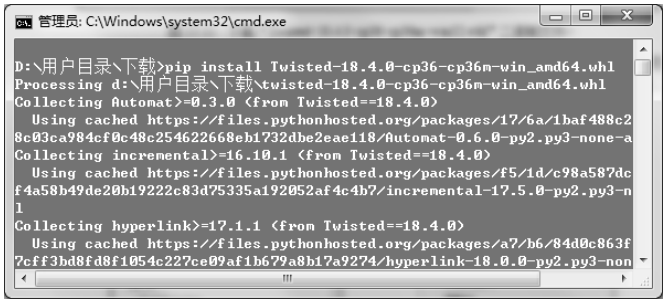


图 16.13　安装 Twisted 模块

2．安装 Scrapy 框架

打开命令行窗口，输入“pip install Scrapy”命令，安装 Scrapy 框架。安装完成以后，在命令行中输入“scrapy”，如果没有出现异常或错误信息，则表示 Scrapy 框架安装成功，如图 16.14 所示。

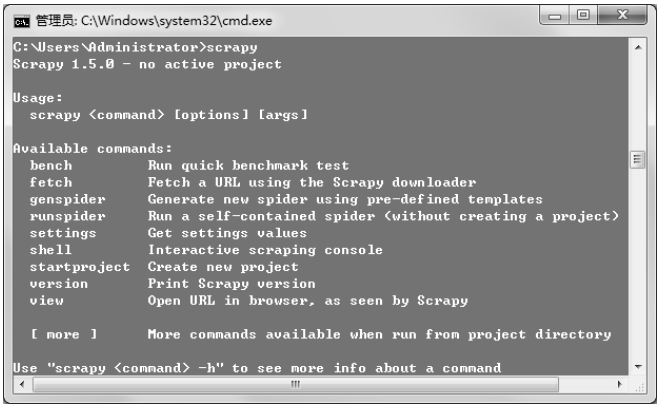

管理员: C:\Windows\system32\cmd.exe
C:\Users\Administrator>scrapy
Scrapy 1.5.0 - no active project
Usage:
scrapy <command> [options] [args]
Available commands:
bench Run quick benchmark test
fetch Fetch a URL using the Scrapy downloader
genspider Generate new spider using pre-defined templates
runspider Run a self-contained spider (without creating a project)
settings Get settings values
shell Interactive scraping console
startproject Create new project
version Print Scrapy version
view Open URL in browser, as seen by Scrapy
[more] More commands available when run from project directory
Use "scrapy <command> -h" to see more info about a command

图 16.14　安装 Scrapy 框架

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **说明：**在安装 Scrapy 框架的过程中，会将 lxml 与 pyOpenSSL 模块同时安装在 Python 环境当中。

3．安装 pywin32 模块

打开命令行窗口，输入“pip install pywin32”命令，安装 pywin32 模块。安装完成以后，在命令行中输入“import pywin32_system32”，如果没有提示错误信息，则表示 pywin32 模块安装成功。

16.4.2 创建 Scrapy 项目

在任意路径下创建一个保存项目的文件夹。例如，在 F:\PycharmProjects 文件夹下打开命令行窗口，输入“scrapy startproject scrapyDemo”命令，即可创建一个名为“scrapyDemo”的 Scrapy 项目，如图 16.15 所示。

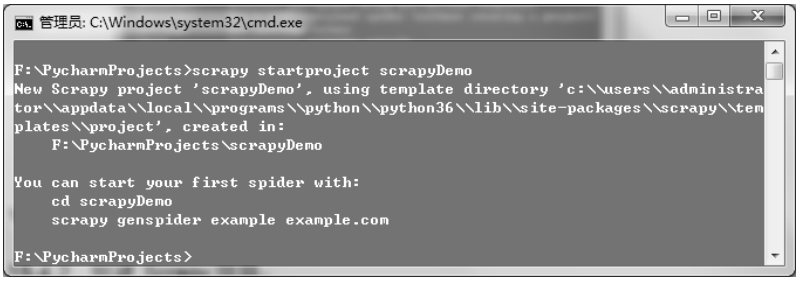


图 16.15　创建一个名为“scrapyDemo”的 Scrapy 项目

为了提高开发效率，笔者使用了第三方开发工具 PyCharm。打开刚刚创建的 scrapyDemo 项目，在左侧的目录结构中可以看到如图 16.16 所示的内容。

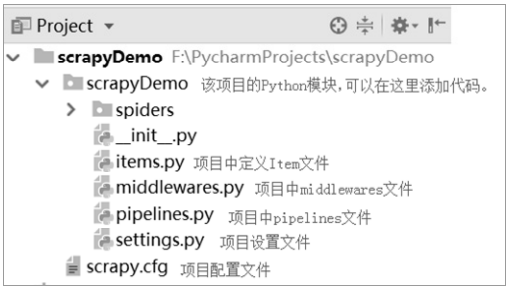


图 16.16　scrapyDemo 项目的目录结构

16.4.3 创建爬虫

在创建爬虫之前，需要先创建一个存放爬虫模块的文件，该文件需要放置在 spiders 文件夹下。爬虫模块是用于从一个或多个网站中爬取数据的类，它需要继承 scrapy.Spider 类。下面通过一个爬虫示例，实现爬取网页后，将网页的代码以 HTML 文件保存至项目文件夹下。示例代码如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
01  import scrapy          # 导入框架
02
03
04 class QuotesSpider(scrapy.Spider):
05     name = "quotes"      # 定义爬虫名称
06
07     def start_requests(self):
08         # 设置爬取目标的地址
09         urls = [
10             'http://quotes.toscrape.com/page/1/',
11             'http://quotes.toscrape.com/page/2/',
12         ]
13         # 获取所有地址，有几个地址发送几次请求
14         for url in urls:
15             # 发送网络请求
16             yield scrapy.Request(url=url, callback=self.parse)
17
18     def parse(self, response):
19         # 获取页数
20         page = response.url.split("/")[-2]
21         # 根据页数设置文件名称
22         filename = 'quotes-%s.html' % page
23         # 以写入文件的模式打开文件。如果没有该文件，则创建该文件
24         with open(filename, 'wb') as f:
25             # 向文件中写入获取的 HTML 代码
26             f.write(response.body)
27         # 输出保存文件的名称
28         self.log('Saved file %s' % filename)
```

在运行 Scrapy 所创建的爬虫项目时，需要在命令行窗口中输入“scrapy crawl quotes”命令，其中“quotes”是自己定义的爬虫名称。由于笔者使用了第三方开发工具 PyCharm，所以需要在底部的 Terminal 窗口中输入运行爬虫的命令行，运行完成以后，将显示如图 16.17 所示的信息。

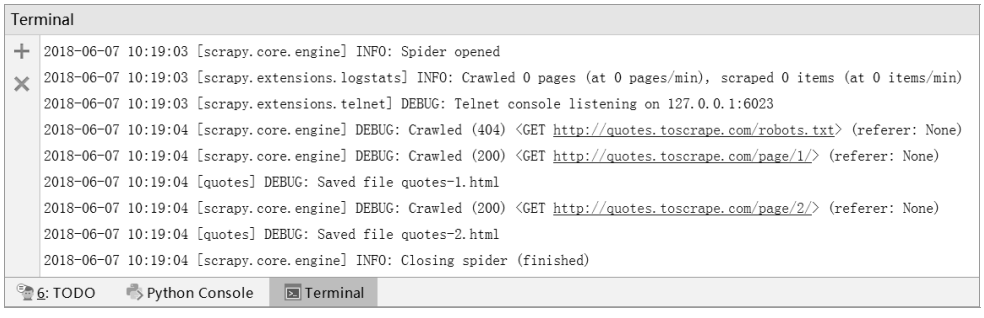


图 16.17　显示启动爬虫后的信息

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

说明：除在命令行窗口中输入命令“scrapy crawl quotes”启动爬虫以外，Scrapy 还提供了可以在程序中启动爬虫的 API，也就是 CrawlerProcess 类。首先在 CrawlerProcess 初始化时传入项目的 settings 信息，然后在 crawl()方法中传入爬虫的名称，最后通过 start()方法启动爬虫。代码如下：

```
# 导入 CrawlerProcess 类
from scrapy.crawler import CrawlerProcess
# 导入获取项目设置信息
from scrapy.utils.project import get_project_settings


# 程序入口
if __name__=='__main__':
    # 创建 CrawlerProcess 类对象并传入项目设置信息参数
    process = CrawlerProcess(get_project_settings())
    # 设置需要启动的爬虫名称
    process.crawl('quotes')
    # 启动爬虫
    process.start()
```

16.4.4 获取数据

Scrapy 爬虫框架可以通过特定的 CSS 或 XPath 表达式来选择 HTML 文件中的某一处，并且提取出相应的数据。CSS（Cascading Style Sheets，层叠样式表）用于控制 HTML 页面布局、字体、颜色、背景及其他效果。XPath 是一门可以在 XML 文档中根据元素和属性查找信息的语言。

1. CSS 提取数据

当使用 CSS 提取 HTML 文件中的某一处数据时，可以指定 HTML 文件中的标签名称。例如，在获取网页的<title>标签中的数据时，可以使用如下代码：

```
response.css('title').extract()
```

获取结果如图 16.18 所示。

```
2018-06-07 11:22:09 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://quotes.toscrape.com/page/1/> (referer: None)
['<title>Quotes to Scrape</title>']
2018-06-07 11:22:09 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://quotes.toscrape.com/page/2/> (referer: None)
['<title>Quotes to Scrape</title>']
```

图 16.18　使用 CSS 提取<title>标签中的数据

说明：返回的内容为 CSS 表达式所对应节点的列表。所以，在提取标签中的数据时，可以使用以下代码：

```
response.css('title::text').extract_first()
```

或者

```
response.css('title::text')[0].extract()
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

2．XPath 提取数据

当使用 XPath 表达式提取 HTML 文件中的某一处数据时，需要根据 XPath 表达式的语法规定来获取指定的数据信息。例如，同样获取<title>标签中的数据，可以使用如下代码：

```
response.xpath('//title/text()').extract_first()
```

下面通过一个示例，实现使用 XPath 表达式获取多条信息。示例代码如下：

```
# 响应信息
def parse(self, response):
    # 获取所有信息
    for quote in response.xpath(".//*[@class='quote']"):
        # 获取名人名言文字信息
        text = quote.xpath(".//*[@class='text']/text()").extract_first()
        # 获取作者
        author = quote.xpath(".//*[@class='author']/text()").extract_first()
        # 获取标签
        tags = quote.xpath(".//*[@class='tag']/text()").extract()
        # 以字典形式输出信息
        print(dict(text=text, author=author, tags=tags))
```

3．翻页提取数据

在以上的示例中，已经实现了获取网页中的数据。如果需要获取整个网页的所有信息，就需要使用翻页功能。例如，获取整个网页的作者名称，可以使用以下代码：

```
# 响应信息
def parse(self, response):
    # div.quote
    # 获取所有信息
    for quote in response.xpath(".//*[@class='quote']"):
        # 获取作者
        author = quote.xpath(".//*[@class='author']/text()").extract_first()
        print(author)  # 输出作者名称

    # 实现翻页
    for href in response.css('li.next a::attr(href)'):
            yield response.follow(href, self.parse)
```

4．创建 Item

爬取网页数据的过程就是从非结构性的数据源中提取结构性数据的过程。例如，在 QuotesSpider 类的 parse()方法中已经获取了 text、author 及 tags 信息，如果需要将这些数据包装成结构化数据，就需要使用 Scrapy 提供的 Item 类。Item 对象是一个简单的容器，用于保存爬取到的数据信息，它提供了一个类似于字典的 API，用于声明其可用字段的便捷语法。Item 使用简单的类定义语法和 Field 对象来声明。在创建 scrapyDemo 项目时，在项目的目录结构中就已经自动创建了一个 items.py 文件，用来定义存储数据信息的 Item 类，它需要继承

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

scrapy.Item。示例代码如下：

```
import scrapy


class ScrapydemoItem(scrapy.Item):
    # define the fields for your item here like:
    # 定义获取的名人名言文字信息
    text = scrapy.Field()
    # 定义获取的作者
    author =scrapy.Field()
    # 定义获取的标签
    tags = scrapy.Field()

    pass
```

Item 创建完成以后，回到自己编写的爬虫代码中，在 parse()方法中创建 Item 对象，然后输出 Item 信息。代码如下：

```
# 响应信息
def parse(self, response):
    # 获取所有信息
    for quote in response.xpath(".//*[@class='quote']"):
        # 获取名人名言文字信息
        text = quote.xpath(".//*[@class='text']/text()").extract_first()
        # 获取作者
        author = quote.xpath(".//*[@class='author']/text()").extract_first()
        # 获取标签
        tags = quote.xpath(".//*[@class='tag']/text()").extract()
        # 创建 Item 对象
        item = ScrapydemoItem(text=text, author=author, tags=tags)
        yield item # 输出 Item 信息
```

> **说明：** Scrapy 爬虫框架的内容较多，这里仅简单地介绍了该爬虫框架的安装、创建爬虫及数据的提取，详细的使用教程可以登录 https://docs.scrapy.org/en/latest 网站查看 Scrapy 的官方文档。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 17 章

Flask Web 框架的使用

如果要进行比较大的 Web 项目的开发，则往往需要花费大量的时间与精力，一次又一次地解决一些相同的问题。本章将介绍使用 Web 框架来解决这类问题。

17.1 Web框架简介

如果你要从零开始建立一些网站，则不得不一次又一次地解决一些相同的问题。这样做是令人厌烦的，并且违背了良好编程的核心原则之一——DRY（不要重复自己）。

有经验的 Web 开发人员在创建新站点时也会遇到类似的问题。当然，总有一些特殊情况会因网站而异，但在大多数情况下，Web 开发人员通常需要处理 4 项任务——数据的创建、读取、更新和删除，也称为 CRUD。幸运的是，通过使用 Web 框架解决了这些问题。

17.1.1 什么是 Web 框架

Web 框架是用来简化 Web 开发的软件框架。框架的存在是为了避免重新造轮子，并且在创建一个新的网站时帮助减少一些开销。典型的框架提供了如下常用功能：

- 管理路由。
- 访问数据库。
- 管理会话和 Cookie。
- 创建模板来显示 HTML。
- 促进代码的重用。

事实上，框架根本不是什么新的东西，它只是一些能够实现常用功能的 Python 文件。我们可以把框架看作工具的集合，而不是特定的东西。框架的存在使得建立网站更快、更容易。此外，框架还促进了代码的重用。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

17.1.2 常用的 Web 框架

前面我们学习了 WSGI（服务器网关接口），它是 Web 服务器和 Web 应用程序或框架之间的一种简单而通用的接口。也就是说，只要遵循 WSGI 接口规则，就可以自主开发 Web 框架。所以，各种开源 Web 框架至少有上百个，关于 Python 框架优劣的讨论也仍在继续。作为初学者，应该选择一些主流的框架来学习使用，这是因为主流框架文档齐全，技术积累较多，社区繁盛，并且能得到更好的支持。下面介绍几种 Python 的主流 Web 框架。

1．Flask

Flask 是一个轻量级的 Web 框架，它的名字暗示了它的含义，它基本上就是一个微型的胶水框架。它把 Werkzeug 和 Jinja 黏合在了一起，所以它很容易被扩展。Flask 有许多的扩展可供用户使用，同时拥有一群忠诚的粉丝和不断增加的用户群。它有一份很完善的文档，甚至还有一份唾手可得的常见范例。Flask 很容易使用，只需要几行代码就可以写出一个“Hello World”程序。

2．Django

这可能是最广为人知和使用最广泛的 Python Web 框架了。Django 有世界上最大的社区和最多的包。它的文档非常完善，并且提供了一站式的解决方案，包括缓存、ORM、管理后台、验证、表单处理等，使得开发复杂的数据库驱动的网站变得简单。但是，Django 系统耦合度较高，替换掉内置的功能比较麻烦。

3．Bottle

这个框架相对来说比较新。Bottle 才是名副其实的微框架——它只有大约 4500 行代码。它除 Python 标准库以外没有任何其他的依赖，甚至还有自己独特的一些模板语言。Bottle 的文档很详细，并且抓住了事物的实质。它很像 Flask，也使用了装饰器来定义路径。

4．Tornado

Tornado 不单单是一个框架，还是一个 Web 服务器。它一开始是给 FriendFeed 开发的，在 2009 年的时候也给 Facebook 使用。它是为了解决实时服务而诞生的。为了做到这一点，Tornado 使用了异步非阻塞 I/O，所以它的运行速度非常快。

以上 4 种框架各有优劣，使用时需要根据自身的应用场景选择适合自己的 Web 框架。本章将着重介绍 Flask 框架。

17.2 Flask框架的使用

Flask 框架依赖两个外部库：Werkzeug 和 Jinja2。Werkzeug 是一个 WSGI（在 Web 应用和多种服务器之间的标准 Python 接口）工具集。Jinja2 负责渲染模板。所以，在安装 Flask 框架之前，需要先安装这两个外部库，最简单的方式就是使用 Virtualenv 创建虚拟环境。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

17.2.1　安装虚拟环境

安装 Flask 框架最便捷的方式是使用虚拟环境。Virtualenv 为每个不同项目提供了一份 Python 安装。它并没有真正安装多个 Python 副本，但是它确实提供了一种巧妙的方式来让各项目环境保持独立。

1．安装 Virtualenv

Virtualenv 的安装非常简单，可以使用如下命令进行安装：

```
pip install virtualenv
```

安装完成后，可以使用如下命令检测 Virtualenv 的版本：

```
virtualenv --version
```

如果运行效果如图 17.1 所示，则说明安装成功。

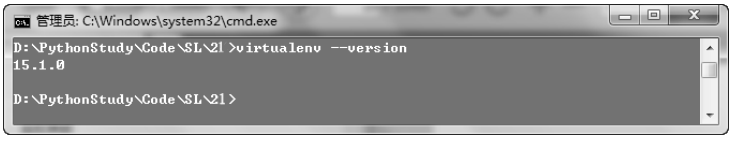


图 17.1　Virtualenv 安装成功

2．创建虚拟环境

使用 Virtualenv 命令在当前文件夹下创建 Python 虚拟环境。这个命令只有一个必需的参数，即虚拟环境的名字。在创建虚拟环境后，在当前文件夹中会出现一个子文件夹，这个子文件夹的名字就是上述命令中指定的参数，与虚拟环境相关的文件都保存在这个子文件夹中。按照惯例，一般虚拟环境会被命名为 venv。运行如下命令：

```
virtualenv venv
```

运行完成后，在运行的目录下，会新增一个 venv 文件夹，它保存一个全新的虚拟环境，其中有一个私有的 Python 解释器，如图 17.2 所示。

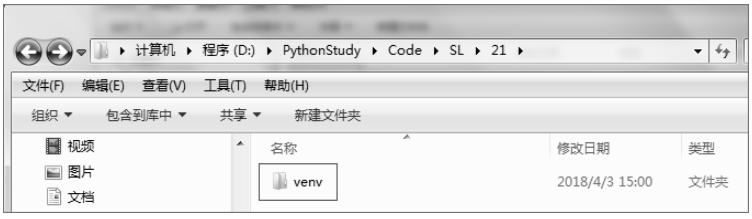


图 17.2　创建虚拟环境

3．激活虚拟环境

在使用这个虚拟环境之前，需要先将其激活。可以通过下面的命令激活这个虚拟环境：

```
venv\Scripts\activate
```

激活以后的效果如图 17.3 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 17.3　激活虚拟环境后的效果

17.2.2　安装 Flask 框架

大多数 Python 包都使用 pip 实用工具安装，在使用 Virtualenv 创建虚拟环境时会自动安装 pip。激活虚拟环境后，pip 所在的路径会被添加到 PATH 中。使用如下命令安装 Flask：

```
pip install flask
```

运行效果如图 17.4 所示。

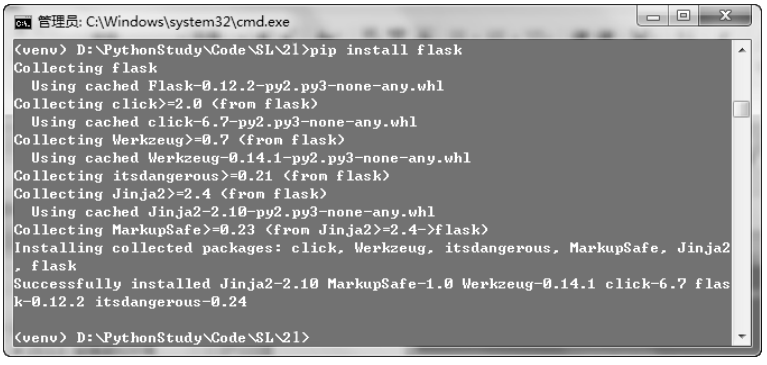


图 17.4　安装 Flask

安装完成以后，可以通过如下命令查看所有的安装包：

```
pip list --format columns
```

运行效果如图 17.5 所示。

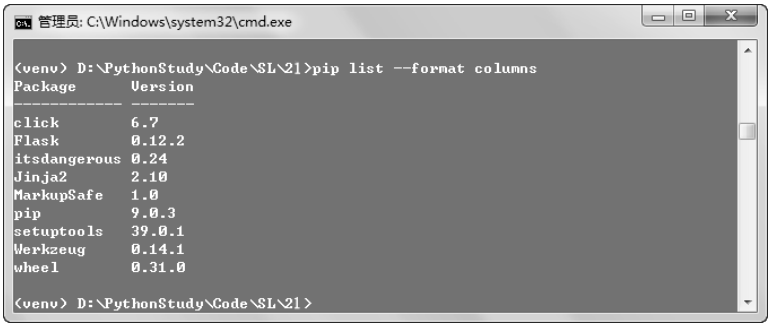


图 17.5　查看所有的安装包

从图 17.5 中可以看到，已经成功安装了 Flask 框架，并且安装了 Flask 框架的两个外部依赖库：Werkzeug 和 Jinja2。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

17.2.3　第一个 Flask 程序

一切准备就绪，现在我们开始编写第一个 Flask 程序。由于是第一个 Flask 程序，所以要从最简单的“Hello World!”开始。

在 venv 同级目录下，创建 01.py 文件，代码如下：

```
01 from flask import Flask
02 app = Flask(__name__)
03
04 @app.route('/')
05 def hello_world():
06     return 'Hello World!'
07 if __name__ == '__main__':
08     app.run()
```

运行 01.py 文件，运行效果如图 17.6 所示。

图 17.6　运行 01.py 文件

然后在浏览器中输入网址“http://127.0.0.1:5000”，运行效果如图 17.7 所示。

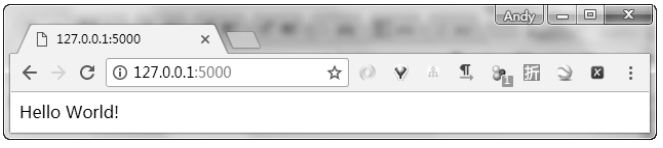


图 17.7　输出“Hello World!”

那么，这段代码做了什么？

- 首先，我们导入 Flask 类，这个类的实例将会是 WSGI 应用程序。
- 接下来，我们创建该类的一个实例，第一个参数是应用模块或者包的名称。如果你使用单一的模块（如本例），则应该使用__name__，因为模块的名称将会因其是作为单独应用启动还是作为模块导入而有所不同（是'__main__'或实际的导入名）。这是必需的，这样 Flask 才知道到哪里去寻找模板、静态文件等。详情见 Flask 的文档。
- 然后，我们使用 app.route 修饰器告诉 Flask 什么样的 URL 能触发函数。
- 这个函数的名字也在生成 URL 时被特定的函数采用，这个函数返回我们想要显示在用户浏览器中的信息。
- 最后，我们使用 run()函数来让应用运行在本地服务器上。其中，if__name__=='__main__':确保服务器只会在该脚本被 Python 解释器直接执行的时候运行，而不会在作为模块被导入的时候运行。

说明：关闭服务器，按<Ctrl+C>组合键。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

17.2.4 开启调试模式

虽然 run()方法适用于启动本地的开发服务器，但是每次修改代码后都要手动重启它，这样并不够优雅，而且 Flask 可以做到更好。如果启用了调试支持，那么服务器会在代码被修改后自动重新载入，并在发生错误时提供一个相当有用的调试器。

有两种途径来开启调试模式。一种途径是直接在应用对象上设置。

```
app.debug = True
app.run()
```

另一种途径是作为 run()方法的一个参数传入。

```
app.run(debug=True)
```

这两种途径的效果完全相同。

17.2.5 路由

客户端（如 Web 浏览器）把请求发送给 Web 服务器，Web 服务器再把请求发送给 Flask 程序实例。因为程序实例需要知道对每个 URL 请求运行哪些代码，所以保存了一个从 URL 到 Python 函数的映射关系。处理 URL 和函数之间关系的程序被称为路由。

在 Flask 程序中定义路由的最简便方式是使用程序实例提供的 app.route 修饰器，把修饰的函数注册为路由。下面的例子说明了如何使用这个修饰器声明路由。

```
01 @app.route('/')
02 def index():
03 return '<h1>Hello World!</h1>'
```

> **说明：**修饰器是 Python 语言的标准特性，可以使用不同的方式修改函数的行为。通常用法是使用修饰器把函数注册为事件的处理程序。

但是，不仅如此！你可以构造含有动态部分的 URL，也可以在一个函数上附着多个规则。

1. 变量规则

要给 URL 添加变量部分，可以把这些特殊的字段标记为<variable_name>，这个部分将会作为命名参数传递到你的函数中。规则可以用<converter:variable_name>指定一个可选的转换器。

创建 02.py 文件，以实例 01 的代码为基础，添加如下代码：

```
01 @app.route('/user/<username>')
02 def show_user_profile(username):
03     # 显示该用户名的用户信息
04     return 'User %s' % username
05
06 @app.route('/post/<int:post_id>')
07 def show_post(post_id):
```


電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    # 根据 ID 显示文章，ID 是整型数据
    return 'Post %d' % post_id
```

在上述代码中使用了转换器，它有下面几种。

- int：接收整数。
- float：同 int，但是接收浮点数。
- path：和默认的相似，但也接收斜线。

运行 02.py 文件，运行效果如图 17.8 和图 17.9 所示。

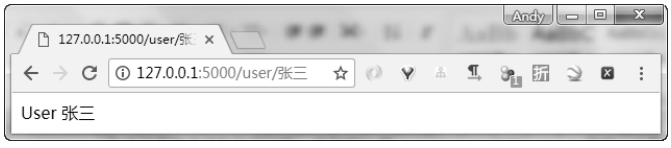


图 17.8　获取用户信息

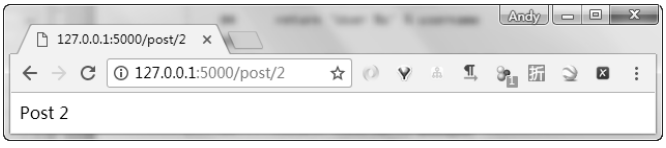


图 17.9　获取文章信息

2．构造 URL

如果 Flask 能匹配 URL，那么 Flask 可以生成它们吗？当然可以。可以用 url_for()来给指定的函数构造 URL。它接收函数名作为第一个参数，也接收对应 URL 规则的变量部分的命名参数。未知变量部分会被添加到 URL 末尾作为查询参数。

创建 03.py 文件，以实例 02 的代码为基础，添加如下代码：

```
from flask import Flask , url_for
app = Flask(__name__)

# 省略其余代码

@app.route('/url/')
def get_url():
    # 根据 ID 显示文章，ID 是整型数据
    return url_for('show_post',post_id=2)

if __name__ == '__main__':
    app.run(debug=True)
```

在上述代码中，设置了“/url/”路由，在访问该路由时，返回“show_post”函数的 URL 信息。运行效果如图 17.10 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

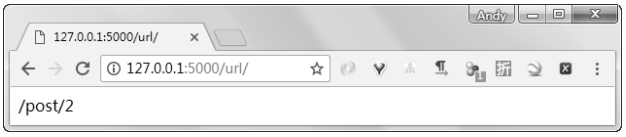


图 17.10　url_for()函数应用效果图

3．HTTP 方法

HTTP（与 Web 应用会话的协议）有许多不同的访问 URL 的方法。在默认情况下，路由只回应 GET 请求，但是通过 app.route 修饰器传递 methods 参数可以改变这个行为。例如：

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        do_the_login()
    else:
        show_the_login_form()
```

HTTP 方法（也经常被叫作“谓词”）告诉服务器，客户端想对请求的页面做一些什么。常用的 HTTP 方法如表 17.1 所示。

表 17.1　常用的 HTTP 方法

方 法 名	说　明
GET	浏览器告诉服务器：只获取页面上的信息并发给我。这是最常用的方法
HEAD	浏览器告诉服务器：欲获取信息，但是只关心消息头。应用应像处理 GET 请求一样来处理它，但是不分发实际内容。在 Flask 中完全无须人工干预，底层的 Werkzeug 库已经替你处理好了
POST	浏览器告诉服务器：想在 URL 上发布新信息。并且，服务器必须确保数据已存储且仅存储一次。这是 HTML 表单常用的向服务器发送数据的方法
PUT	类似 POST，但是服务器可能触发了存储过程多次，多次覆盖掉旧值。你可能会问这有什么用，当然这是有原因的。考虑到传输中连接可能会丢失，在这种情况下，浏览器和服务器之间的系统可能第二次安全地接收请求，而不破坏其他东西。因为 POST 只触发一次，所以用 POST 是不可能的
DELETE	删除给定位置的信息
OPTIONS	给客户端提供一个敏捷的途径来弄清楚这个 URL 支持哪些 HTTP 方法。从 Flask 0.6 开始，实现了自动处理

17.2.6　静态文件

动态 Web 应用也会需要静态文件，通常是 CSS 和 JavaScript 文件。在理想状况下，已经配置好 Web 服务器来提供静态文件，但是在开发中，Flask 也可以做到。只要在你的包中或模块所在的目录中创建一个名为 static 的文件夹，在应用中使用/static 即可访问。

给静态文件生成 URL，使用特殊的“static”端点名。

```
url_for('static', filename='style.css')
```

这个静态文件应该存储在文件系统上的 static/style.css 中。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

17.2.7　模板

模板是一个包含响应文本的文件，其中包含用占位变量表示的动态部分，其具体值只在请求的上下文中才能知道。先使用真实值替换变量，再返回最终得到的响应字符串，这一过程被称为渲染。为了渲染模板，Flask 使用了一个名为 Jinja2 的强大模板引擎。

1．渲染模板

在默认情况下，Flask 在程序文件夹的 templates 子文件夹中寻找模板。下面通过一个示例学习如何渲染模板。

在 venv 同级目录下创建 templates 文件夹，然后创建两个文件并分别命名为 index.html 和 user.html。在 venv 同级目录下创建 04.py 文件，渲染这些模板。目录结构如图 17.11 所示。

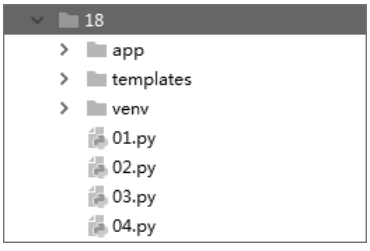


图 17.11　目录结构

Templates/index.html 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
    <h1>Hello World!</h1>
</body>
</html>
```

Templates/user.html 代码如下：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <h1>Hello, {{ name }}!</h1>
</body>
</html>
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

04.py 代码如下：

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return render_template('index.html')

@app.route('/user/<username>')
def show_user_profile(username):
    # 显示该用户名的用户信息
    return render_template('user.html', name=name)

if __name__ == '__main__':
    app.run(debug=True)
```

Flask 提供的 render_template()函数把 Jinja2 模板引擎集成到了程序中。render_template()函数的第一个参数是模板的文件名；随后的参数都是键值对，表示模板中变量对应的真实值。在这段代码中，第二个模板收到一个名为 name 的变量。上述代码中的 name=name 是关键字参数，这类关键字参数很常见，但如果你不熟悉它们，则可能会觉得迷惑且难以理解。左边的“name”表示参数名，就是模板中使用的占位符；右边的“name”是当前作用域中的变量，表示同名参数的值。

2．变量

在模板中使用的{{ name }}结构表示一个变量，它是一种特殊的占位符，告诉模板引擎这个位置的值是从渲染模板时所使用的数据中获取的。Jinja2 能识别所有类型的变量，甚至一些复杂的类型，如列表、字典和对象。在模板中，一些使用变量的示例如下：

```
<p>从字典中取一个值: {{ mydict['key'] }}.</p>
<p>从列表中取一个值: {{ mylist[3] }}.</p>
<p>从列表中取一个带索引的值: {{ mylist[myintvar] }}.</p>
<p>从对象的方法中取一个值: {{ myobj.somemethod() }}.</p>
```

可以使用过滤器修改变量，过滤器名添加在变量名之后，中间使用竖线分隔。例如，下述模板以首字母大写形式显示变量 name 的值。

```
Hello, {{ name|capitalize }}
```

Jinja2 提供的常用过滤器如表 17.2 所示。

表 17.2　Jinja2 提供的常用过滤器

名　称	说　明
safe	渲染值时不转义
capitalize	把值的首字母转换成大写形式，把其他字母转换成小写形式
lower	把值转换成小写形式

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

名　称	说　明
upper	把值转换成大写形式
title	把值中每个单词的首字母都转换成大写形式
trim	把值的首尾空格去掉
striptags	在渲染之前，把值中所有的 HTML 标签删掉

safe 过滤器值得特别说明一下。在默认情况下，出于安全考虑，Jinja2 会转义所有变量。例如，如果一个变量的值为'<h1>Hello</h1>'，那么 Jinja2 会将其渲染成'<h1>Hello</h1>'，浏览器能显示这个 h1 元素，但不会进行解释。在很多情况下，需要显示变量中存储的 HTML 代码，这时就可以使用 safe 过滤器。

3．控制结构

Jinja2 提供了多种控制结构，可用来改变模板的渲染流程。下面使用简单的例子来介绍其中最有用的控制结构。

下面这个例子展示了如何在模板中使用条件控制语句。

```
{% if user %}
Hello, {{ user }}!
{% else %}
Hello, Stranger!
{% endif %}
```

一种常见需求是在模板中渲染一组元素。下面的例子展示了如何使用 for 循环实现这一需求。

```
<ul>
{% for comment in comments %}
<li>{{ comment }}</li>
{% endfor %}
</ul>
```

Jinja2 还支持宏，宏类似于 Python 代码中的函数。例如：

```
{% macro render_comment(comment) %}
<li>{{ comment }}</li>
{% endmacro %}
<ul>
{% for comment in comments %}
{{ render_comment(comment) }}
{% endfor %}
</ul>
```

为了重复使用宏，我们可以将其保存在单独的文件中，然后在需要使用的模板中导入。

```
{% import 'macros.html' as macros %}
<ul>
{% for comment in comments %}
{{ macros.render_comment(comment) }}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
05 {% endfor %}
06 </ul>
```

需要在多处重复使用的模板代码片段可以写入单独的文件中，再包含在所有模板中，以避免重复。

```
{% include 'common.html' %}
```

另一种重复使用代码的强大方式是模板继承，它类似于 Python 代码中的类继承。创建一个名为 base.html 的基模板。

```
01 <html>
02 <head>
03 {% block head %}
04 <title>{% block title %}{% endblock %} - My Application</title>
05 {% endblock %}
06 </head>
07 <body>
08 {% block body %}
09 {% endblock %}
10 </body>
11 </html>
```

block 标签定义的元素可在衍生模板中被修改。在本例中，我们定义了名为 head、title 和 body 的块。注意，title 块包含在 head 块中。下面这个示例是基模板的衍生模板。

```
01 {% extends "base.html" %}
02 {% block title %}Index{% endblock %}
03 {% block head %}
04 {{ super() }}
05 <style>
06 </style>
07 {% endblock %}
08 {% block body %}
09 <h1>Hello, World!</h1>
10 {% endblock %}
```

extends 指令声明这个模板衍生自 base.html。在 extends 指令之后，基模板中的 3 个块被重新定义，模板引擎会将其插入适当的位置。注意新定义的 head 块，在基模板中其内容不是空的，所以使用 super()函数获取原来的内容。

17.3 小结

本章主要介绍了 Python 的 Web 开发框架。首先介绍为什么要使用 Web 框架，以及 Python 中常用的 Web 框架。然后重点介绍 Flask 框架，主要包括框架的安装、基础知识和基本使用等内容。通过对本章的学习，读者会对 Flask 框架有基本的了解，为以后使用 Flask 框架开发项目打下良好的基础。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 5 篇　Web 开发篇

第 18 章

中华诗词大会（网络版）（Django 框架+Bootstrap+MySQL+Redis 实现）

18.1　项目概述

本章将介绍使用 Django 框架实现中华诗词大会（网络版）的开发。该系统的主要功能是批量上传题目、参与者答题、排行榜排名等。该项目使用的 Django 版本是 1.11.2，数据库版本是 MySQL 5.7，此外还使用了 Redis 数据库。通过对这个项目的学习，希望读者可以提高对 Django 框架的掌握能力。

18.2　系统功能设计

18.2.1　了解项目整体开发流程

在开发项目之前，首先需要对项目的整体开发流程有一个熟悉的过程。项目整体开发流程如图 18.1 所示。

从图 18.1 中可以看出，一个项目有很多角色的参与，比如客户、市场人员、产品经理、项目经理、开发团队、测试团队等。而作为一名开发人员，最主要的工作是项目的开发，也就是图 18.1 中方框标注的部分。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

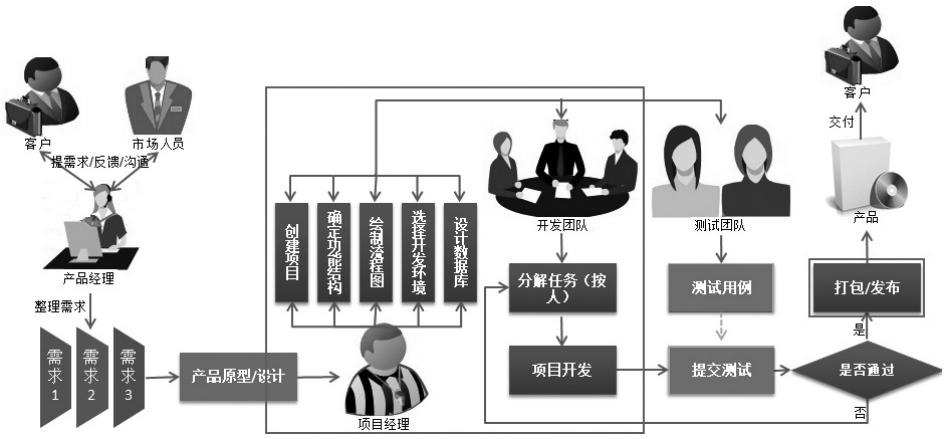


图 18.1　项目整体开发流程

18.2.2　系统功能结构

中华诗词大会（网络版）的功能结构如图 18.2 所示。

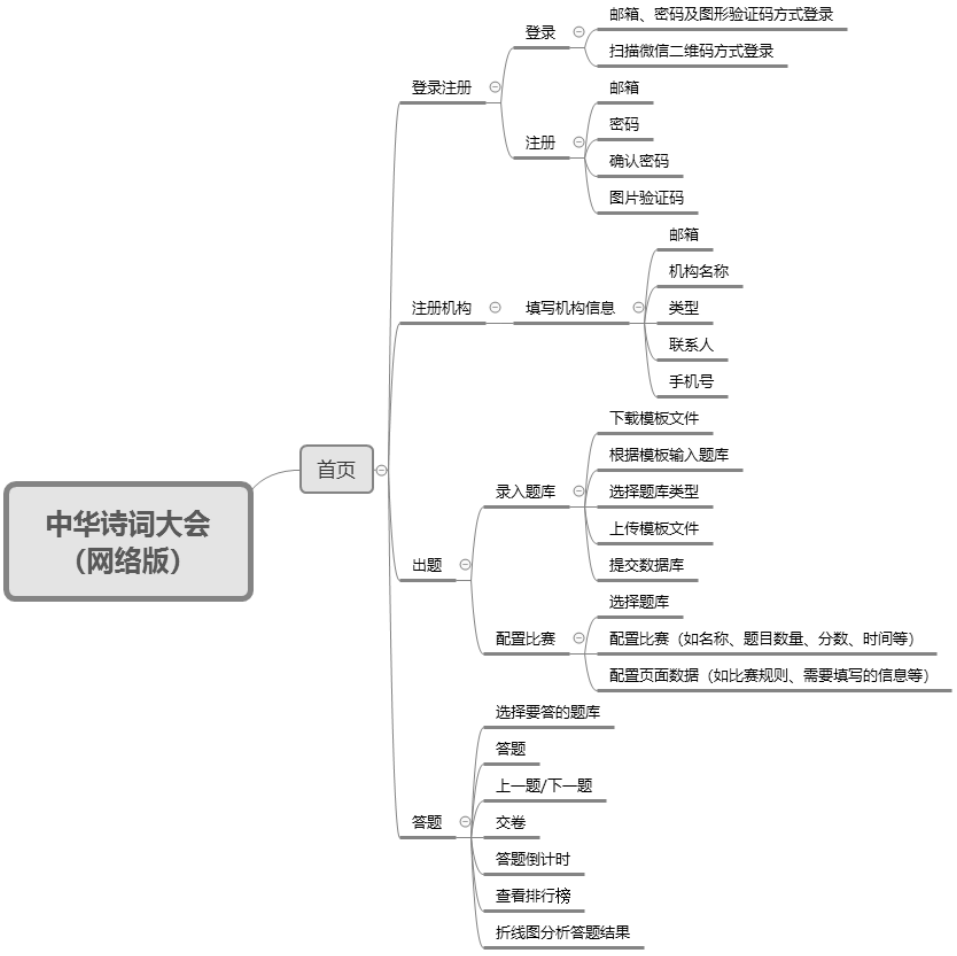


图 18.2　中华诗词大会（网络版）的功能结构

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

18.2.3　系统业务流程

中华诗词大会（网络版）的业务流程如图 18.3 所示。

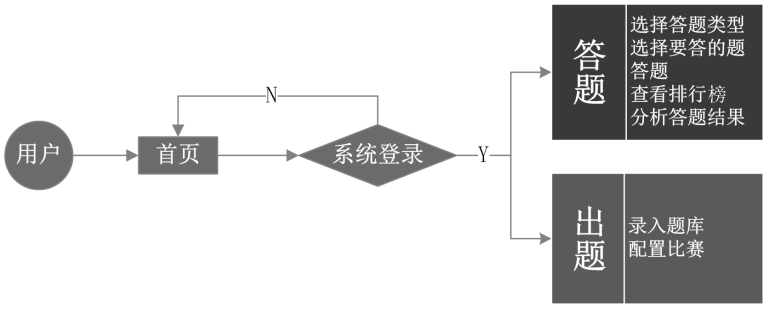


图 18.3　中华诗词大会（网络版）的业务流程

18.2.4　系统预览

中华诗词大会（网络版）是一个答题、出题一站式管理平台，该系统中包含很多页面，下面展示几个比较重要的页面。

中华诗词大会（网络版）首页如图 18.4 所示。

图 18.4　首页

中华诗词大会（网络版）出题页面如图 18.5 所示。

图 18.5　出题页面

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

中华诗词大会（网络版）答题页面如图 18.6 所示。

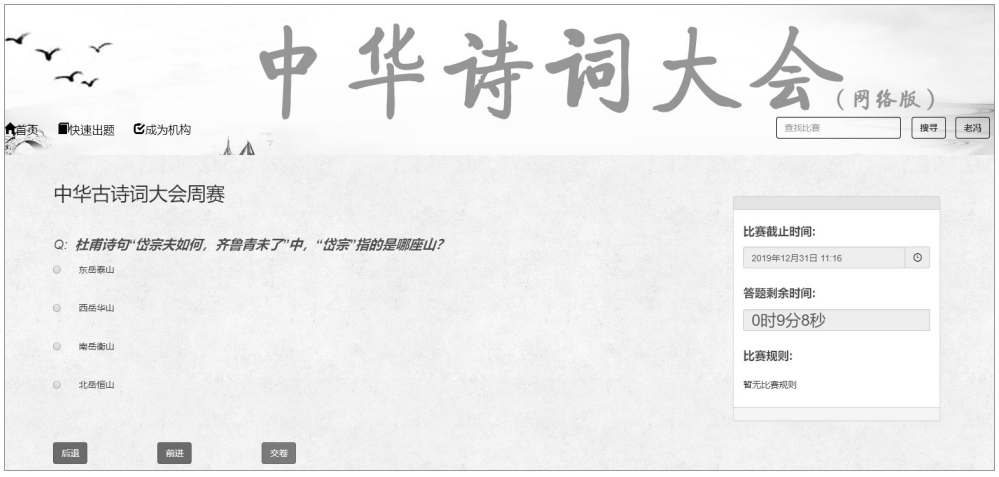


图 18.6　答题页面

中华诗词大会（网络版）答题分析页面如图 18.7 所示。

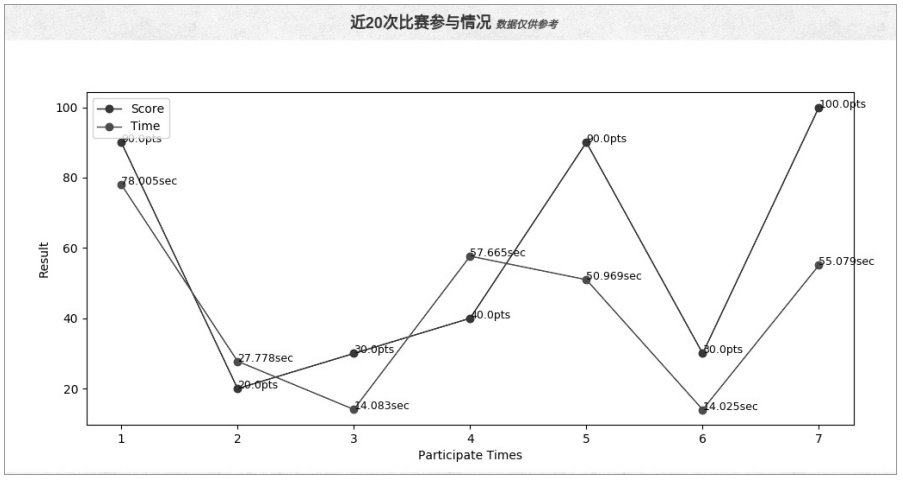


图 18.7　答题分析页面

18.3　系统开发必备

18.3.1　系统开发环境

本系统的软件开发及运行环境具体如下。

- 操作系统：Windows 7、Windows 8、Windows 10 或 Ubuntu。
- 虚拟环境：Virtualenv 或 Anaconda。
- 数据库和驱动：MySQL + PyMySQL、Redis。
- 开发工具：PyCharm。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- 开发框架：Django + Bootstrap + jQuery。
- 浏览器：Chrome。

18.3.2　项目目录结构

中华诗词大会（网络版）项目目录结构如图 18.8 所示。

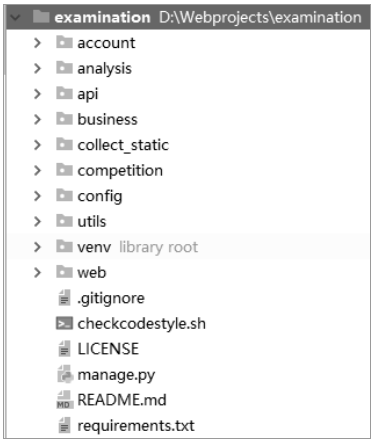


图 18.8　项目目录结构

图 18.8 列出了中华诗词大会（网络版）的项目目录结构，该结构中的文件夹及文件的作用分别如下。

- account：配置用户属性和用户信息数据的 App，其中的视图包含登录视图和首页渲染视图。
- analysis：包含数据分析、埋点和图表生成视图的 App。
- api：RESTfull API 的接口路由包，不包含视图。
- business：机构账户、应用配置和固定的额外配置数据 App，包含机构数据渲染的视图。
- collect_static：Django 的 STATIC_ROOT 目录，用来配置 nginx 路由的目录。
- competition：核心比赛功能 App，包含比赛数据渲染和答题信息、录入题库等接口视图。
- config：项目配置文件目录，包含公共配置文件、本地配置文件和数据库配置文件等。
- utils：包含封装后的 MySQL 模块、扩展的 Redis 接口、修饰器、封装的响应类、中间件、题库上传工具、微信登录类和错误码等工具。
- venv：Virtualenv 项目虚拟环境包。
- web：项目前端代码。
- .gitignore、README.md、LICENSE：项目代码版本控制的配置文件。
- checkcodestyle.sh：在 Shell 下检查代码 pep8 规范和执行 isort.py 工具的脚本。
- requirements.txt：项目所依赖的 Python 包的 pip 安装列表。
- manage.py：Django 命令入口。

中华诗词大会（网络版）使用 Django 框架进行开发，该框架中的 manage.py 文件提供了众多管理命令接口，方便执行数据库迁移和静态资源收集等操作。本项目中使用的主要命令如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
python manage.py makemigrations  # 生成数据库迁移脚本
python manage.py migrate  # 根据 makemigrations 命令生成的脚本，创建或修改数据库表结构
python manage.py migrate migrate_name  # 回滚到指定迁移版本
python manage.py collectstatic  # 生成静态资源目录,根据 settings.py 中的 STATIC_ROOT 设置
python manage.py shell          # 打开 Django 解释器，可以引入项目包
python manage.py dbshell        # 打开 Django 数据库连接，可以执行原生 SQL 命令
python manage.py startproject  # 创建一个 Django 项目
python manage.py startapp       # 创建一个 App
python manage.py createsuperuser # 创建一个管理员超级用户，使用 django.contrib.auth 认证
python manage.py runserver      # 运行开发服务器
```

18.4 数据库设计

18.4.1 数据库概要说明

中华诗词大会（网络版）使用 MySQL 数据库来存储数据，数据库名为 exam，共包含 22 张数据表，其结构如图 18.9 所示。

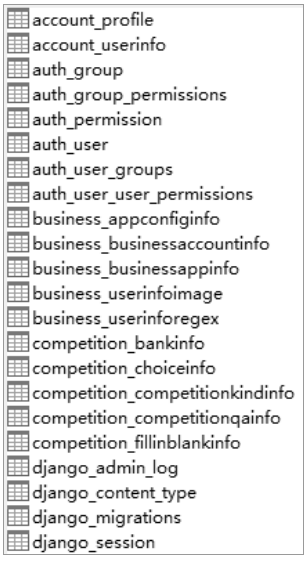


图 18.9　数据库表结构

exam 数据库中的数据表对应的中文表名及其作用如表 18.1 所示。

表 18.1　exam 数据库中的数据表对应的中文表名及其作用

英文表名	中文表名	作　　用
account_profile	用户信息表	保存授权后的账户信息
account_userinfo	用户填写信息表	保存用户填写的表单信息
auth_group	授权组表	保存 Django 默认的授权组信息

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

英文表名	中文表名	作　用
auth_group_permissions	授权组权限表	保存 Django 默认的授权组权限信息
auth_permission	授权权限表	保存 Django 默认的权限信息
auth_user	授权用户表	保存 Django 默认的用户授权信息
auth_user_groups	授权用户组表	保存 Django 默认的用户组信息
auth_user_user_permissions	授权用户权限表	保存 Django 默认的用户权限信息
business_appconfiginfo	机构 App 配置表	保存机构 App 配置信息
business_businessaccountinfo	机构账户表	保存机构账户信息
business_businessappinfo	机构 App 表	保存机构 App 信息，与配置信息关联
business_userinfoimage	表单图片链接表	保存每个表单字段的图片链接
business_userinforegex	表单验证正则表	保存每个表单字段的正则表达式信息
competition_bankinfo	题库信息表	保存题库信息
competition_choiceinfo	选择题表	保存选择题信息
competition_competitionkindinfo	比赛信息表	保存比赛信息和比赛配置信息
competition_competitionqainfo	答题记录表	保存答题记录
competition_fillinblankinfo	填空题表	保存填空题信息
django_admin_log	Django 日志表	保存 Django 管理员登录日志
django_content_type	Django contenttype 表	保存 Django 默认的 contenttype
django_migrations	Django 迁移表	保存 Django 的数据库迁移记录
django_session	Django session 表	保存 Django 默认的授权等 session 记录

18.4.2　数据表模型

Django 框架自带的 ORM 可以满足绝大多数数据库开发的需求，在没有达到一定的数量级时，我们完全不需要担心 ORM 为项目带来的瓶颈。下面是中华诗词大会（网络版）中使用 ORM 来管理比赛信息的数据模型，代码如下：

```
from django.db import models
from django.utils import timezone
from django.utils.translation import ugettext_lazy as _

from shortuuidfield import ShortUUIDField
from utils.basemodels import CreateUpdateMixin, MediaMixin


class CompetitionKindInfo(CreateUpdateMixin):
    """比赛类别信息类"""

    CATE0 = 0
    CATE1 = 1
    CATE2 = 2
    CATE3 = 3
    CATE4 = 4
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    CATE5 = 5

    KIND_TYPES = (
        (CATE0, u'综合类'),
        (CATE1, u'叙事类'),
        (CATE2, u'抒情类'),
        (CATE3, u'山水类'),
        (CATE4, u'咏物类'),
        (CATE5, u'送别类'),
    )

    kind_id = ShortUUIDField(_(u'比赛 id'), max_length=32, blank=True, null=True,
help_text=u'比赛类别唯一标识', db_index=True)
    account_id = models.CharField(_(u'出题账户 id'), max_length=32, blank=True,
null=True, help_text=u'商家账户唯一标识', db_index=True)
    app_id = models.CharField(_(u'应用 id'), max_length=32, blank=True, null=True,
help_text=u'应用唯一标识', db_index=True)
    bank_id = models.CharField(_(u'题库 id'), max_length=32, blank=True, null=True,
help_text=u'题库唯一标识', db_index=True)
    kind_type = models.IntegerField(_(u'比赛类型'), default=CATE0, choices=KIND_TYPES,
help_text=u'比赛类型')
    kind_name = models.CharField(_(u'比赛名称'), max_length=32, blank=True,
null=True, help_text=u'竞赛类别名称')

    sponsor_name = models.CharField(_(u'赞助商名称'), max_length=60, blank=True,
null=True, help_text=u'赞助商名称')

    total_score = models.IntegerField(_(u'总分数'), default=0, help_text=u'总分数
')
    question_num = models.IntegerField(_(u'题目个数'), default=0, help_text=u'出
题数量')

    # 周期相关
    cop_startat = models.DateTimeField(_(u'比赛开始时间'), default=timezone.now,
help_text=_(u'比赛开始时间'))
    period_time = models.IntegerField(_(u'答题时间'), default=60, help_text=u'答
题时间(min)')
    cop_finishat = models.DateTimeField(_(u'比赛结束时间'), blank=True, null=True,
help_text=_(u'比赛结束时间'))

    # 参与相关
    total_partin_num = models.IntegerField(_(u'total_partin_num'), default=0,
help_text=u'总参与人数')

    class Meta:
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        verbose_name = _(u'比赛类别信息')
        verbose_name_plural = _(u'比赛类别信息')

    def __unicode__(self):
        return str(self.pk)

    @property
    def data(self):
        return {
            'account_id': self.account_id,
            'app_id': self.app_id,
            'kind_id': self.kind_id,
            'kind_type': self.kind_type,
            'kind_name': self.kind_name,
            'total_score': self.total_score,
            'question_num': self.question_num,
            'total_partin_num': self.total_partin_num,
            'cop_startat': self.cop_startat,
            'cop_finishat': self.cop_finishat,
            'period_time': self.period_time,
            'sponsor_name': self.sponsor_name,
        }
```

18.5　用户登录模块设计

18.5.1　用户登录模块概述

用户登录模块主要对进入中华诗词大会（网络版）的用户信息进行验证。本项目中提供了两种登录方式，分别是使用邮箱和密码的方式，以及微信扫码登录的方式。使用邮箱和密码方式登录的效果如图 18.10 所示；微信扫码登录的效果如图 18.11 所示。

图 18.10　使用邮箱和密码方式登录

图 18.11　微信扫码登录

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

18.5.2 使用 Django 默认的用户授权机制实现普通登录

Django 默认的用户授权机制可以提供绝大多数场景的登录功能。为了更加适应答题系统的需求，这里对其进行简单修改。

1. 用户登录接口

在 account app 目录下创建一个 login_views.py 文件，用来作为接口视图。在该文件中编写一个 normal_login()方法，用来实现用户正常的用户名和密码登录功能。代码如下：

```
import base64                                                    # base64 加密模块
import uuid                                                      # uuid 模块
from io import BytesIO                                           # 流传输模块
from django.conf import settings                                 # 项目配置模块
from django.contrib.auth import authenticate, login              # 引入授权和登录方法
from django.contrib.auth.models import User                      # 引入用户数据模型
from django.core.mail import send_mail                           # 引入发送邮件方法
from django.db import transaction                                # 引入数据库事务模块
from django.shortcuts import redirect                            # 引入重定向方法
from django.views.decorators.csrf import csrf_exempt             # 引入防跨站请求伪造装饰器
from account.models import Profile                               # 引入 Profile 数据模型
from furl import furl                                            # 引入 furl 模块
from utils.codegen import get_pic_code                           # 引入生成验证码方法
from utils.errors import UserError                               # 引入错误信息类
from utils.redis.extensions import quote, unquote                # 引入 Redis 扩展
from utils.redis.rprofile import (get_vcode, set_profile, set_signcode, set_vcode,
                    set_passwd, set_has_sentemail, get_has_ sentemail,
                    set_has_sentregemail, get_has_sentregemail) # 引入 Redis 函数
from utils.response import json_response                         # 引入响应对象
# 微信登录模块
from utils.wechat_utils import AUTHORIZE_URI, get_access_info, get_userinfo
@csrf_exempt                                                     # POST 表单防止跨站请求伪造
@transaction.atomic                                              # 数据库操作失败回滚
def normal_login(request):
    email = request.POST.get('email', '')                        # 获取 E-mail
    password = request.POST.get('password', '')                  # 获取 password
    sign = request.POST.get('sign', '')                          # 获取登录验证码的 sign
    vcode = request.POST.get('vcode', '')                        # 获取用户输入的验证码
    result = get_vcode(sign)  # 在 Redis 中校验 sign 和 vcode
    if not (result and (result.decode('utf-8') == vcode.lower())):
        return json_response(*UserError.VeriCodeError) # 校验失败返回错误码 300003
    try:
        user = User.objects.get(email=email)                     # 使用 E-mail 获取 Django 用户
    except User.DoesNotExist:
        return json_response(*UserError.UserNotFound)  # 获取失败返回错误码 300001
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    # 授权校验
    user = authenticate(request, username=user.username, password=password)
    if user is not None:                                    # 校验成功，获得返回用户信息
        login(request, user)                                # 登录用户，设置登录 session
        # 获取或创建 Profile 数据
        profile, created = Profile.objects.select_for_update().get_or_create(
            email=user.email,
        )
        if profile.user_src != Profile.COMPANY_USER:
            profile.name = user.username
            profile.user_src = Profile.NORMAL_USER
            profile.save()
        request.session['uid'] = profile.uid                # 设置Profile uid的session
        request.session['username'] = profile.name          # 设置用户名的 session
        set_profile(profile.data)  # 将用户信息保存到 Redis 中，用户信息在 Redis 中查询
    else:
        return json_response(*UserError.PasswordError) # 校验失败,返回错误码 300002
    return json_response(200, 'OK', {  # 返回 JSON 格式的数据
        'user_info': profile.data,
        'has_login': bool(profile),
    })
```

以上实现的是用户普通登录的接口。编写完上述代码后，需要在 api 模块下的 urls.py 文件中添加路由，代码如下：

```
url(r'^login_normal$', login_views.normal_login, name='normal_login'),
```

在 web 目录下的 base.html 文件中定义一个使用 jQuery 实现的 AJAX 异步请求方法，用来处理用户登录的表单。代码如下：

```
$('#signInNormal').click(function () {
    refreshVcode('signin');
    $('#signInModalNormal').modal('show');
    $('#signInVcodeImg').click(function () {
        refreshVcode('signin');
    });
    $('#signInPost').click(function () {
        var email = $('#signInId').val();
        var password = $('#signInPassword').val();
        var vcode = $('#signInVcode').val();
        if(!checkEmail(email)){
            $('#signInId').val('');
            $('#signInId').attr('placeholder', '邮件格式错误');
            $('#signInId').css('border', '1px solid red');
            return false;
        }else{
            $('#signInId').css('border', '1px solid #C1FFC1');
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        }
        if(!password){
            $('#signInPassword').attr('placeholder', '请填写密码');
            $('#signInPassword').css('border', '1px solid red');
        }else{
            $('#signInPassword').css('border', '1px solid #C1FFC1');
        }
        $.ajax({
        url: '/api/login_normal',
        data: {
            'email': email,
            'password': password,
            'sign': loginSign,
            'vcode': vcode
        },
        type: 'post',
        dataType: 'json',
        success: function(res){
            if (res.status === 200){
                $('#signInModalNormal').modal('hide');
                window.location.href = '/';
            }
            else if(res.status === 300001) {
                alert('用户名错误');
            }
            else if(res.status === 300002) {
                alert('密码错误');
            }
            else if(res.status === 300003) {
                alert('验证码错误');
            }
            else {
                alert('登录错误');
            }
        }
    })
    });
```

登录使用异步方式来实现。当用户单击页面上的“登录”按钮时，弹出 Bootstrap 框架的 modal 插件。当用户输入邮箱账户、密码和验证码时，会根据不同的错误信息给用户一个友好的提示。

当前端验证全部通过后，AJAX 发起请求，后台会校验用户输入的数据是否合理、有效。如果验证全部通过，则将在用户单击“登录”按钮时，显示存储在 session 中的用户名。用户登录页面如图 18.12 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

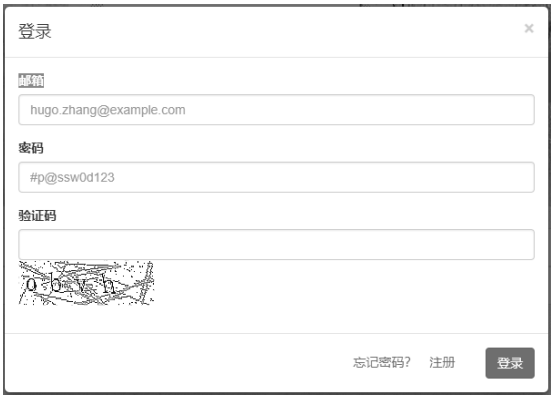


图 18.12　用户登录页面

> **说明：**在登录过程中需要刷新验证码，我们为此也提供了一个接口。在本项目中，通过创建 utils/codegen.py/CodeGen 类，来实现验证码生成和保存到流的过程。具体代码请查看资源包中的源码文件。

2．用户注册接口

用户注册同样使用 AJAX 异步请求的方式，在弹出的 modal 中输入表单内容，然后通过正则表达式规则进行校验，如果校验成功，则将输入提交到后台进行校验，如果校验通过，则将返回一个新渲染的视图，并提示用户发送邮件去验证邮箱。发送邮件同样需要通过异步请求的接口实现。用户注册的视图函数代码如下：

```
@csrf_exempt
@transaction.atomic
def signup(request):
    email = request.POST.get('email', '')                       # 邮箱
    password = request.POST.get('password', '')                 # 密码
    password_again = request.POST.get('password_again', '')     # 确认密码
    vcode = request.POST.get('vcode', '')                       # 注册验证码
    sign = request.POST.get('sign')             # 注册验证码校验位
    if password != password_again:              # 两次密码不一样，返回错误码 300002
        return json_response(*UserError.PasswordError)
    result = get_vcode(sign)                    # 校验 vcode，逻辑和登录视图相同
    if not (result and (result.decode('utf-8') == vcode.lower())):
        return json_response(*UserError.VeriCodeError)
    if User.objects.filter(email__exact=email).exists(): # 检查数据库中是否存在该用户
        return json_response(*UserError.UserHasExists)    # 返回错误码 300004
    username = email.split('@')[0]              # 生成一个默认的用户名
    if User.objects.filter(username__exact=username).exists():
        username = email                        # 默认用户名已存在，使用邮箱作为用户名
    User.objects.create_user(                   # 创建用户，并设置为不可登录
        is_active=False,
        is_staff=False,
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        username=username,
        email=email,
        password=password,
    )
    Profile.objects.create(                    # 创建用户信息
        name=username,
        email=email
    )
    sign = str(uuid.uuid1())                   # 生成邮箱校验码
    set_signcode(sign, email)                  # 在 Redis 中设置 30min 时限的验证周期
    return json_response(200, 'OK', {          # 返回 JSON 格式的数据
        'email': email,
        'sign': sign
    })
```

编写完上面的视图函数后，需要在 api 模块下的 urls.py 文件中加入如下路由：

```
url(r'^signup$', login_views.signup, name='signup'),
```

响应接口数据后，注册过程并未完成，需要用户手动触发邮箱验证。当用户单击“发送邮件”按钮时，AJAX 将会提交数据到接口路由。代码如下：

```
url(r'^sendmail$', login_views.sendmail, name='sendmail'),
```

上面路由对应的视图函数为 sendmail()，该函数仅仅完成了一个使用 django.core.sendmail 发送邮件的过程。其实现代码如下：

```
def sendmail(request):
    to_email = request.GET.get('email', '')# 在 URL 中获取的注册邮箱地址
    sign = request.GET.get('sign', '')     # 在 URL 中获取的 sign 标识
    # 检查用户是否在同一时间多次单击“发送邮件”按钮
    if not get_has_sentregemail(to_email):
        title = '[Quizz.cn 用户激活邮件]'      # 定义邮件标题
        sender = settings.EMAIL_HOST_USER   # 获取发送邮件的邮箱地址
       url = settings.DOMAIN + '/auth/email_notify?email=' + to_email + '&sign=' +
sign                                        # 回调函数
        # 邮件内容
        msg = '您好，Quizz.cn 管理员想邀请您激活您的用户，点击链接激活。{}'.format(url)
        # 发送邮件并获取发送结果
        ret = send_mail(title, msg, sender, [to_email], fail_silently=True)
        if not ret:
            # 发送出错，返回错误码 300006
            return json_response(*UserError.UserSendEmailFailed)
        set_has_sentregemail(to_email)      # 正常发送，设置 3 分钟的继续发送限制
        return json_response(200, 'OK', {}) # 返回空的 JSON 格式的数据
    else:
        # 如果用户在同一时间多次单击“发送邮件”按钮，则返回错误码 300005
        return json_response(*UserError.UserHasSentEmail)
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **说明：**在上面发送邮件的视图函数 sendmail()中添加了一个回调函数，用来检查用户是否确认邮件。这个回调函数是一个普通的视图渲染函数。

在 config 模块下的 urls.py 文件中添加总的授权路由，代码如下：

```
urlpatterns += [
    url(r'^auth/', include('account.urls', namespace='auth')),  # 授权
]
```

然后在 account 模块下的 urls.py 文件中添加授权回调函数的路由，代码如下：

```
url(r'^email_notify$', login_render.email_notify, name='email_notify'),
```

授权回调函数 email_notify()的实现代码如下：

```
@transaction.atomic
def email_notify(request):
    email = request.GET.get('email', '')   # 获取要验证的邮箱
    sign = request.GET.get('sign', '')     # 获取校验码
    signcode = get_signcode(sign)          # 在 Redis 中校验邮箱
    if not signcode:
        return render(request, 'err.html', VeriCodeTimeOut)# 校验失败返回错误视图
    if not (email == signcode.decode('utf-8')):
        return render(request, 'err.html', VeriCodeError)  # 校验失败返回错误视图
    try:
        user = User.objects.get(email=email)        # 获取用户
    except User.DoesNotExist:
        user = None
    if user is not None:                            # 激活用户
        user.is_active = True
        user.is_staff = True
        user.save()
        login(request, user)                        # 登录用户
        # 配置用户信息
        profile, created = Profile.objects.select_for_update().get_or_create(
            name=user.username,
            email=user.email,
        )
        profile.user_src = Profile.NORMAL_USER      # 配置用户为普通登录用户
        profile.save()

        request.session['uid'] = profile.uid        # 配置 session
        request.session['username'] = profile.name
        return render(request, 'web/index.html', { # 渲染视图，并返回已登录信息
            'user_info': profile.data,
            'has_login': True,
            'msg': "激活成功",
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        })
    else:
        return render(request, 'err.html', VerifyFailed)  # 校验失败返回错误视图
```

前端单击注册链接的 AJAX 请求代码如下：

```
$('#signUpPost').click(function () {
    var email = $('#signUpId').val();
    var password = $('#signUpPassword').val();
    var passwordAgain = $('#signUpPasswordAgain').val();
    var vcode = $('#signUpVcode').val();
    if(!checkEmail(email)) {
        $('#signUpId').val('');
        $('#signUpId').attr('placeholder', '邮箱格式错误');
        $('#signUpId').css('border', '1px solid red');
        return false;
    }else{
        $('#signUpId').css('border', '1px solid #C1FFC1');}
    if(!(password === passwordAgain)) {
        $('#signUpPasswordAgain').val('');
        $('#signUpPasswordAgain').attr('placeholder', '两次密码输入不一致');
        $('#signUpPassword').css('border', '1px solid red');
        $('#signUpPasswordAgain').css('border', '1px solid red');
        return false;
    }else{
        $('#signUpPassword').css('border', '1px solid #C1FFC1');
        $('#signUpPasswordAgain').css('border', '1px solid #C1FFC1');}
    $.ajax({
        url: '/api/signup',
        type: 'post',
        data: {
            'email': email,
            'password': password,
            'password_again': passwordAgain,
            'sign': loginSign,
            'vcode': vcode},
        dataType: 'json',
        success: function (res) {
            if(res.status === 200) {
                sign = res.data.sign;
                email = res.data.email;
                window.location.href = '/auth/signup_redirect?email=' + email +
'&sign=' + sign;
            }else if(res.status === 300002) {
                alert('两次输入密码不一致');
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            }else if(res.status === 300003) {
                alert('验证码错误');
            }else if(res.status === 300004) {
                alert('用户名已存在');
            }
        }
    })
});
```

发送邮件的 AJAX 请求代码如下：

```
$('#sendMail').click(function () {
    $('#sendMailLoading').modal('show');
   $.ajax({
       url: '/api/sendmail',
       type: 'get',
       data: {
           'email': '{{ email|safe }}',
           'sign': '{{ sign|safe }}'
       },
       dataType: 'json',
       success: function (res) {
           if(res.status === 200) {
               $('#sendMailLoading').modal('hide');
               alert('发送成功，快去登录邮箱激活账户吧');
           }
           else if(res.status === 300005) {
               $('#sendMailLoading').modal('hide');
               alert('您已经发送过邮件，请稍等再试');
           }
           else if(res.status === 300006) {
               $('#sendMailLoading').modal('hide');
               alert('验证邮件发送失败!');
           }
       }
   })
});
```

说明：修改密码和重置密码的实现方式与用户注册的实现方式类似，这里不再赘述。

用户注册页面如图 18.13 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 18.13　用户注册页面

18.5.3　基于 Oauth 2.0 的微信登录的实现

Oauth 2.0 是开放授权的一个标准，允许用户授权第三方网站访问他们存储在应用提供方服务器上面的信息，而这个过程不需要用户提供用户名和密码给第三方网站，也不允许第三方网站分享他们在应用服务器上面的所有数据内容。类似于 Google、Facebook、Twitter、新浪微博、微信和 QQ 等都提供了 Oauth 2.0 的授权平台。在开发项目时，如果要在网站上添加微信授权登录功能，则可以使用扫描网页二维码的方式实现快速授权登录。

本节主要讲解中华诗词大会（网络版）如何实现微信扫码登录功能。以下是前端代码。当用户单击“微信登录”按钮时，弹出一个 Bootstrap modal，并调用微信官方提供的授权二维码生成接口。

```
$('#signInQrcode').click(function () {
    $('#signInModal').modal('show');
    var obj = new WxLogin({
        id: "signInModal",
        appid: "",
        scope: "snsapi_login",
        redirect_uri: "",
        state: "webwxlogin",
        style: "white",
        href: ""
    });
});
```

在 api 模块下的 urls.py 文件中添加一条 API 路由，代码如下：

```
url(r'^login_redirect$', login_views.login_redirect, name='index'),
```

然后在 account app 目录下的 login_views.py 文件中添加 login_redirect()回调函数，代码如下：

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
@transaction.atomic
def login_redirect(request):
    code = request.GET.get('code', '')      # 获取授权类型
    state = request.GET.get('state', '')    # 获取 state 参数，用来回调校验
    state = unquote_state(request, state)  # 将 state 参数中的 Unicode 字符反编码
    # 获取 access_token
    access_info = get_access_info(settings.WXWEB_LOGIN_PARAMS.get('appid', ''),
settings.WXWEB_LOGIN_PARAMS.get('appsecret', ''), code)
    if 'errcode' in access_info:            # 获取失败，重定向到重新授权页面
        return
redirect(AUTHORIZE_URI.format(settings.WXWEB_LOGIN_PARAMS.get('appid'),
settings.WEB_LOGIN_REDIRECT_URI, 'snsapi_userinfo', state))
    # 获取用户信息
    userinfo = get_userinfo(access_info.get('access_token', ''), access_info.get
('openid', ''))
    if 'openid' not in userinfo:            # 未获取到 open_id，重定向到重新获取页面
        return
redirect(AUTHORIZE_URI.format(settings.WXWEB_LOGIN_PARAMS.get('appid'),
settings.WEB_LOGIN_REDIRECT_URI, 'snsapi_userinfo', state))
    # 根据获取的用户信息填充我们自己的数据库
    profile_values = {
        'openid': userinfo.get('openid', ''),
        'user_src': 1,
        'sex': userinfo.get('sex', 0),
        'nickname': userinfo.get('nickname', ''),
        'avatar': userinfo.get('headimgurl', ''),
        'country': userinfo.get('country', ''),
        'province': userinfo.get('province', ''),
        'city': userinfo.get('city', ''),
    }
    # 获取更新或者创建用户
    profile, created = Profile.objects.select_for_update().get_or_create(openid=
userinfo.get('openid', ''), defaults=profile_values)
    if not profile.unionid:
        profile.unionid = userinfo.get('unionid', '')
        profile.save()
    # 未创建，defaults 参数不会传入 get_or_create()方法中，所以要更新 defaults 里面所有的值
    if not created:
        for key, value in profile_values.items():
            setattr(profile, key, value)
        profile.save()
    # 保存用户信息到 Redis 中
    set_profile(profile.data)
    # 将登录结果写到 session 中
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    request.session['uid'] = profile.uid
    request.session['username'] = profile.name
    # 因为我们将授权之后的跳转 URL 保存到 state 参数里，所以登录成功后，重定向到此 URL 即可
    return redirect(furl(state).url)
```

使用微信等 Oauth 2.0 授权的方法非常简单，这种机制可以让我们很方便地进行用户管理。而且对于用户来说，他们的操作要比普通的用户名和密码登录更快捷。微信扫码登录的效果如图 18.14 所示。

图 18.14　微信扫码登录

> **说明：**在实现微信扫码登录功能时，需要开发者在微信开放平台中注册“网站应用程序”，待微信团队审核通过后，开发者得到 AppID，然后即可通过 AppID 进行微信扫码登录功能的开发。

18.5.4　机构注册功能的实现

在中华诗词大会（网络版）中还提供了机构注册功能。当用户单击“成为机构”导航按钮时，需要根据用户的 uid 来判断用户是否已经注册过机构账户。如果没有注册过，则渲染一个表单，这个表单使用 AJAX 来异步请求；如果已经注册过，则返回一个信息提示，引导用户重定向到出题页面。下面详细讲解实现过程。

在 config/urls.py 中添加机构 App 的路由，代码如下：

```
url(r'^biz/', include('business.urls', namespace='biz')),  # 机构
```

然后在 business 下面的 urls.py 文件中添加渲染机构页面的路由，代码如下：

```
url(r'^$', biz_render.home, name='index'),
```

在上面的代码中用到了页面渲染视图函数，函数名称为 index，其具体实现如下：

```
import time
from django.shortcuts import render              # 导入 render 函数
from django.contrib.auth.models import User      # 导入 User 模型类
from business.models import BusinessAccountInfo # 导入 BusinessAccountInfo 模型类
from account.models import Profile               # 导入 Profile 模型类
def home(request):
    uid = request.GET.get('uid', '')             # 获取 uid
    try:
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        profile = Profile.objects.get(uid=uid)          # 根据 uid 获取用户信息
    except Profile.DoesNotExist:
        profile = None  # 未获取到用户信息，将 profile 变量置空
    types = dict(BusinessAccountInfo.TYPE_CHOICES)      # 所有的机构类型
    # 渲染视图，返回机构类型和是否存在该账户绑定过的机构账户
    return render(request, 'bussiness/index.html', {
        'types': types,
        'is_company_user': bool(profile) and (profile.user_src == Profile.COMPANY_USER)
    })
```

在 web/business/index.html 页面中添加一个 Bootstrap 框架的 panel 控件，用来存放机构注册表单。代码如下：

```
<div class="panel panel-info">
        <div class="panel-heading"><h3 class="panel-title">注册成为机构</h3></div>
        <div class="panel-body">
            <form id="bizRegistry" class="form-group">
                <label for="bizEmail">邮箱</label>
                <input type="text" class="form-control" id="bizEmail" placeholder=
"填写机构邮箱" />
                <label for="bizCompanyName">名称</label>
                <input   type="text"   class="form-control"   id="bizCompanyName"
placeholder="填写机构名称" />
                <label for="bizCompanyType">类型</label>
                <select id="bizCompanyType" class="form-control">
                    {% for k, v in types.items %}
                        <option value="{{ k }}">{{ v }}</option>
                    {% endfor %}
                </select>
                <label for="bizUsername">联系人</label>
                <input type="text" class="form-control" id="bizUsername" placeholder=
"填写机构联系人" />
                <label for="bizPhone">手机号</label>
                <input type="text" class="form-control" id="bizPhone" placeholder=
"填写联系人手机" />
                <input type="submit" id="bizSubmit" class="btn btn-primary" value=
"注册机构" style="float: right;margin-top: 20px" />
            </form>
        </div>
    </div>
</div>
```

在 JavaScript 脚本中添加申请成为机构的请求方法，代码如下：

```
$('#bizSubmit').click(function () {
    var email = $('#bizEmail').val();
    var name = $('#bizCompanyName').val();
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    var type = $('#bizCompanyType').val();
    var username = $('#bizUsername').val();
    var phone = $('#bizPhone').val();
    if(!email.match('^\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*$')) {
        $('#bizEmail').val('');
        $('#bizEmail').attr('placeholder', '邮箱格式错误');
        $('#bizEmail').css('border', '1px solid red');
        return false;
    }else{
        $('#bizEmail').css('border', '1px solid #C1FFC1');
    }
    if(!(name.match('^[a-zA-Z0-9_\\u4e00-\\u9fa5]{4,20}$'))) {
        $('#bizCompanyName').val('');
        $('#bizCompanyName').attr('placeholder', '请填写由 4～20 位中文、字母、数字或
者下画线组成的机构名称');
        $('#bizCompanyName').css('border', '1px solid red');
        return false;
    }else{
        $('#bizCompanyName').css('border', '1px solid #C1FFC1');
    }
    if(!(username.match('^[\u4E00-\u9FA5A-Za-z]+$'))){
        $('#bizUsername').val('');
        $('#bizUsername').attr('placeholder', '联系人姓名应该为汉字或大小写字母');
        $('#bizUsername').css('border', '1px solid red');
        return false;
    }else{
        $('#bizUsername').css('border', '1px solid #C1FFC1');
    }
    if(!(phone.match('^1[3|4|5|8][0-9]\\d{4,8}$'))){
        $('#bizPhone').val('');
        $('#bizPhone').attr('placeholder', '手机号不符合规则');
        $('#bizPhone').css('border', '1px solid red');
        return false;
    }else{
        $('#bizPhone').css('border', '1px solid #C1FFC1');
    }
    $.ajax({
        url: '/api/checkbiz',
        type: 'get',
        data: {
            'email': email
        },
        dataType: 'json',
        success: function (res) {
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            if(res.status === 200) {
                if(res.data.bizaccountexists) {
                    alert('您的账户已存在，请直接登录');
                    window.location.href = '/';
                }
                else if(res.data.userexists && !res.data.bizaccountexists) {
                    if(confirm('您的邮箱已被注册为普通用户，我们将会为您绑定该用户。')){
                      bizPost(email, name, type, username, phone, 1);
                      window.location.href = '/biz/notify?email=' + email + '&bind=1';
                    }else {
                       window.location.href = '/{% if request.session.uid %}?uid=
{{ request.session.uid }}{% else %}{% endif %}';
                    }
                }
                else{
                    bizPost(email, name, type, username, phone, 2);
                    window.location.href = '/biz/notify?email=' + email;
                }
            }
        }
    });
    function bizPost(email, name, type, username, phone, flag) {
        $.ajax({
            url: '/api/regbiz',
            data: {
                'email': email,
                'name': name,
                'type': type,
                'username': username,
                'phone': phone,
                'flag': flag
            },
            type: 'post',
            dataType: 'json'
        })
    }
});
```

在单击“注册”按钮时，首先验证表单是否符合正则表达式。当这些验证都通过后，先请求一个/api/check_biz 接口。这个接口对应的路由和接口函数如下：

```
def check_biz(request):
    email = request.GET.get('email', '')  # 获取邮箱
    try:  # 检查数据库中是否有该邮箱注册过的数据
        biz = BusinessAccountInfo.objects.get(email=email)
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    except BusinessAccountInfo.DoesNotExist:
        biz = None
    return json_response(200, 'OK', {  # 返回是否已经被注册过和是否已经有此用户
        'userexists': User.objects.filter(email=email).exists(),
        'bizaccountexists': bool(biz)
    })
```

上面的接口用来检查用户填写的邮箱是否存在登录账户和机构账户，这里的实现是：如果用户的登录账户存在，但是机构账户不存在，那么会提示用户绑定已有账户，注册成为机构账户；如果用户的登录账户不存在，并且机构账户也不存在，则会在请求下一个接口时，为该邮箱创建一个未激活的登录账户和一个机构账户,所以该用户务必去自己的邮箱里面验证该账户并激活。

那么，如果是上面所说的第一种情况，就没必要再次验证邮箱了。

而如果是第二种情况，也就是用户既没有注册为登录用户，也没有注册机构账户的情况，那么用户会将表单信息提交到/api/regbiz 接口。因此，首先在 api 模块下的 urls.py 文件中添加路由，代码如下：

```
# bussiness
urlpatterns += [
    url(r'^regbiz$', biz_views.registry_biz, name='registry biz'),
    url(r'^checkbiz$', biz_views.check_biz, name='check_biz'),
]
```

然后在 business 模块下的 biz_views.py 文件中添加如下代码：

```
@csrf_exempt
@transaction.atomic
def registry_biz(request):
    email = request.POST.get('email', '')  # 获取填写的邮箱
    name = request.POST.get('name', '')    # 获取填写的机构名
    username = request.POST.get('username', '')  # 获取填写的机构联系人
    phone = request.POST.get('phone', '')  # 获取填写的手机号
    ctype = request.POST.get('type', BusinessAccountInfo.INTERNET)  # 获取机构类型
    flag = int(request.POST.get('flag', 2))# 获取一个标记位，代表用户是创建新用户还是使用绑定老用户的方式
    uname = email.split('@')[0]    # 和之前的注册逻辑没什么区别，创建一个账户名
    if not User.objects.filter(username__exact=name).exists():
        final_name = username
    elif not User.objects.filter(username__exact=uname).exists():
        final_name = uname
    else:
        final_name = email
    if flag == 2:                  # 如果标记位是 2，那么将为他创建新用户
        user = User.objects.create_user(
            username=final_name,
```


電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            email=email,
            password=settings.INIT_PASSWORD,
            is_active=False,
            is_staff=False
        )
    if flag == 1:                # 如果标记位是 1，那么将为他绑定老用户
        try:
            user = User.objects.get(email=email)
        except User.DoesNotExist:
            return json_response(*UserError.UserNotFound)
    pvalues = {
        'phone': phone,
        'name': final_name,
        'user_src': Profile.COMPANY_USER,
    }
    # 获取或创建用户信息
    profile, _ = Profile.objects.select_for_update().get_or_create(email=email)
    for k, v in pvalues.items():
        setattr(profile, k, v)
    profile.save()
    bizvalues = {
        'company_name': name,
        'company_username': username,
        'company_phone': phone,
        'company_type': ctype,
    }
    # 获取或创建机构账户信息
    biz, _ = BusinessAccountInfo.objects.select_for_update().get_or_create(
        email=email,
        defaults=bizvalues
    )
    # 响应 JSON 格式的数据，这个标记位在发送验证邮件的时候还有用
    return json_response(200, 'OK', {
        'name': final_name,
        'email': email,
        'flag': flag
    })
```

在提交表单后，如果是新创建的用户，则需要验证用户的邮箱，这个步骤和之前雷同，所以不再赘述。整个过程完成后，如果用户注册成为机构账户，那么他可以在快速出题的导航页中录制题库，并且生成一个比赛。登录账户会根据注册渠道的不同，被标记为普通用户、微信登录用户和机构账户用户 3 种类型。机构注册页面的效果如图 18.15 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 18.15　机构注册页面

18.6　核心答题功能的设计

18.6.1　答题首页设计

答题首页主体呈现为若干个导航标签和每个标签对应的比赛列表，其效果如图18.16所示。

图 18.16　答题首页导航标签及对应的比赛列表

在 base.html 中，定义两个模板标签，代码如下：

```
<title>{% block title %}{% endblock %}</title>      # 用来展示网页标题
{% block content %}
{% endblock %}                                      # 用来放置每个子页面的主体内容
```

其他所有的页面都扩展自 base.html，如比赛首页的 HTML 代码如下：

```
{% extends '../base.html' %}
{% block title %}比赛首页{% endblock %}
{% block content %}
{% endblock %}
```

在上面的代码中，在 content 标签中填写比赛首页的主体内容，extends 用来描述所继承的基础页面。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

比赛首页放置在 web/web/index.html 文件中，当单击每个比赛类型标签时，都会触发一个 AJAX 请求，该请求会通过一个/api/games/s/cate0 接口获取该类型下的热门比赛，具体如下。

- hot：代表所有热门比赛前十位。
- cate0：代表综合类比赛前十位。
- cate1：代表叙事类古诗词比赛前十位。
- cate2：代表抒情类古诗词比赛前十位。
- cate3：代表山水类古诗词比赛前十位。
- cate4：代表咏物类古诗词比赛前十位。
- cate5：代表送别类古诗词比赛前十位。

下面以 hot 类型（热门比赛）为例进行讲解。首先编写单击热门比赛标签时调用的 JavaScript 函数，代码如下：

```
$(document).ready(function () {    # 页面加载，即默认加载热门比赛
    $('#hot').addClass('btn btn-primary');
    loadHot();
});
$('#hot').click(function () {      # 单击“热门比赛”按钮加载热门比赛
    removeClass();
    $(this).addClass('btn btn-primary');
    loadHot();
});
function loadHot() {
    $.ajax({
        url: '/api/games/s/hot',
        data: {},
        type: 'get',
        dataType: 'json',
        success: function (res) {
            kinds = res.data.kinds;
            var strHTML = '';
            if(!kinds.length){$('#kindList').html("<div class=\"alert alert-danger\">
<strong>暂无热门比赛</strong></div>")}else{
                for(var i in kinds){
                    strHTML += "<div id=\"gameList\" title=\"举办机构:";
                    strHTML += kinds[i].sponsor_name ? kinds[i].sponsor_name : "暂无";
                    strHTML += "\" data-toggle=\"tooltip\" data-placement=\"left\">";
                    strHTML += "<a href=\"{% url 'bs:index' %}?uid={{ user_info.
uid }}&kind_id=" + kinds[i].kind_id + "\" class=\"list-group-item\"><strong>" +
kinds[i].kind_name + "</strong><span class=\"badge\"><small style=\"color:
yellow\">热度:</small> " + kinds[i].total_partin_num + "</span></a>\n";
                    strHTML += "</div>";}
                $('#kindList').html(strHTML);
}}})}
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

在 api 模块下的 urls.py 文件中添加路由，代码如下：

```
url(r'^games/s/(\w+)$', game_views.games, name='query_games'),
```

上述代码中用到了 games()视图函数，该函数用来获取指定类型的比赛，其实现代码如下：

```
@csrf_exempt
@check_login
def games(request, s):
    if s == 'hot':          # 获取前十名热门比赛
        kinds = CompetitionKindInfo.objects.filter(
            cop_finishat__gt=datetime.datetime.now(tz=datetime.timezone.utc),
        ).order_by('-total_partin_num').order_by('-created_at')[:10]
    elif s == 'cate0':          # 获取所有综合类比赛
        kinds = CompetitionKindInfo.objects.filter(
            kind_type=CompetitionKindInfo.CATE0,
            cop_finishat__gt=datetime.datetime.now(tz=datetime.timezone.utc)
        ).order_by('-total_partin_num').order_by('-created_at')[:10]
    elif s == 'cate1':          # 获取所有叙事类古诗词比赛
        kinds = CompetitionKindInfo.objects.filter(
            kind_type=CompetitionKindInfo.CATE1,
            cop_finishat__gt=datetime.datetime.now(tz=datetime.timezone.utc)
        ).order_by('-total_partin_num').order_by('-created_at')[:10]
    elif s == 'cate2':      # 获取所有抒情类古诗词比赛
        kinds = CompetitionKindInfo.objects.filter(
            kind_type=CompetitionKindInfo.CATE2,
            cop_finishat__gt=datetime.datetime.now(tz=datetime.timezone.utc)
        ).order_by('-total_partin_num').order_by('-created_at')[:10]
    elif s == 'cate3':      # 获取所有山水类古诗词比赛
        kinds = CompetitionKindInfo.objects.filter(
            kind_type=CompetitionKindInfo.CATE3,
            cop_finishat__gt=datetime.datetime.now(tz=datetime.timezone.utc)
        ).order_by('-total_partin_num').order_by('-created_at')[:10]
    elif s == 'cate4':      # 获取所有咏物类古诗词比赛
        kinds = CompetitionKindInfo.objects.filter(
            kind_type=CompetitionKindInfo.CATE4,
            cop_finishat__gt=datetime.datetime.now(tz=datetime.timezone.utc)
        ).order_by('-total_partin_num').order_by('-created_at')[:10]
    elif s == 'cate5':  # 获取所有送别类古诗词比赛
        kinds = CompetitionKindInfo.objects.filter(
            kind_type=CompetitionKindInfo.CATE5,
            cop_finishat__gt=datetime.datetime.now(tz=datetime.timezone.utc)
        ).order_by('-total_partin_num').order_by('-created_at')
    else:
        kinds = None         # 标签错误不返回
    return json_response(200, 'OK', {  # 返回 JSON 格式的数据和比赛列表
        'kinds': [i.data for i in kinds]
    })
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

每个比赛类型下面的所有比赛列表都对应着一个比赛详情页面的 URL。比赛首页的 URL 需要在 config/urls.py 中添加一个总的比赛 URL，代码如下：

```
url(r'^$', login_render.web_index, name='web_index'),
```

比赛详情页面也要对应一套 urls，在 config/urls.py 中进行配置，代码如下：

```
url(r'^bs/', include('competition.urls', namespace='bs')),  # 比赛
```

然后在 competition app 下面添加一个 urls.py 文件，并添加一条子路由，代码如下：

```
url(r'^$', cop_render.home, name='index'),
```

18.6.2　比赛详情页面

比赛详情页面用来展示比赛的信息，包括比赛名称、出题机构、题目数量和题库大小等，其效果如图 18.17 所示。

图 18.17　比赛详情页面

在 competition app 下面添加一个 cop_render.py 文件，用来存放比赛页面的视图渲染函数，代码如下：

```
import collections                          # 导入第三方 collections 模块
import json                                 # 导入 JSON 模块
from django.shortcuts import render         # 导入 render 方法
from account.models import Profile          # 导入 Profile 数据模型
# 导入比赛类数据模型
from competition.models import BankInfo, CompetitionKindInfo, CompetitionQAInfo
from utils.decorators import check_copstatus, check_login  # 导入登录检查修饰器
from utils.errors import (BankInfoNotFound, CompetitionNotFound, ProfileNotFound,
                          QuestionLogNotFound, QuestionNotSufficient) # 导入错误视图信息
# 导入页面配置 Redis 信息
from utils.redis.rpageconfig import get_pageconfig, get_form_regex
from utils.redis.rprofile import get_enter_userinfo    # 导入用户 Redis 信息
from utils.redis.rrank import get_rank, get_rank_data # 导入排行榜等 Redis 信息
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
@check_login
def home(request):
    uid = request.GET.get('uid', '')                    # 获取 uid
    kind_id = request.GET.get('kind_id', '')            # 获取 kind_id
    created = request.GET.get('created', '0')           # 获取标志位，以后会用到
    try:  # 获取比赛数据
        kind_info = CompetitionKindInfo.objects.get(kind_id=kind_id)
    except CompetitionKindInfo.DoesNotExist:            # 不存在渲染错误视图
        return render(request, 'err.html', CompetitionNotFound)
    try:  # 获取题库数据
        bank_info = BankInfo.objects.get(bank_id=kind_info.bank_id)
    except BankInfo.DoesNotExist:                       # 不存在渲染错误视图
        return render(request, 'err.html', BankInfoNotFound)
    try:  # 获取用户数据
        profile = Profile.objects.get(uid=uid)
    except Profile.DoesNotExist:                        # 不存在渲染错误视图
        return render(request, 'err.html', ProfileNotFound)
    # 题目数量是否小于题库大小
    if kind_info.question_num > bank_info.total_question_num:
        return render(request, 'err.html', QuestionNotSufficient)
    # 从 Redis 中获取页面配置信息
    show_info = get_pageconfig(kind_info.app_id).get('show_info', {})
    # 页面配置信息，用来控制答题前是否展示一张表单
    is_show_userinfo = show_info.get('is_show_userinfo', False)
    form_fields = collections.OrderedDict()# 生成一个有序的、用来保存表单字段的字典
    form_regexes = []                       # 生成一个空的正则表达式列表
    if is_show_userinfo:
        # 从页面配置中获取 userinfo_fields
        userinfo_fields = show_info.get('userinfo_fields', '').split('#')
        for i in userinfo_fields: # 将页面配置的每个正则表达式取出来，放入正则表达式列表中
            form_regexes.append(get_form_regex(i))
        userinfo_field_names = show_info.get('userinfo_field_names', '').split('#')
        # 将每个表单字段信息保存到有序的表单字段字典中
        for i in range(len(userinfo_fields)):
            form_fields.update({userinfo_fields[i]: userinfo_field_names[i]})
    return render(request, 'competition/index.html', {  # 渲染页面
        'user_info': profile.data,
        'kind_info': kind_info.data,
        'bank_info': bank_info.data,
        'is_show_userinfo': 'true' if is_show_userinfo else 'false',
        'userinfo_has_enterd': 'true' if get_enter_userinfo(kind_id, uid) else
'false',
        'userinfo_fields': json.dumps(form_fields) if form_fields else '{}',
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        'option_fields': json.dumps(show_info.get('option_fields', '')),
        'field_regexes': form_regexes,
        'created': created
    })
```

比赛详情页面除了返回比赛的信息，还需要返回页面的配置信息。在本项目中，在 business app 的数据模型中创建一个 AppConfigInfo，关联每个 BusinessAppInfo 的 app_id，用来指定每个 AppInfo 在页面中的不同配置，以便于让整个页面多样化、可定制化。在这里我们指定了一个配置，如果机构账户开启了此功能，那么每个答题用户需要在参与比赛之前填写一张表单，如图 18.18 所示。

图 18.18　答题之前需要填写的表单

图 18.18 所示的表单主要是为了收集答题用户的信息，以便于日后联系该用户。在 business.models 模块中，添加一个名为 AppConfigInfo 的模型类，代码如下：

```
class AppConfigInfo(CreateUpdateMixin):
    app_id = models.CharField(_(u'应用 id'), max_length=32, help_text=u'应用唯一标识', db_index=True)# 应用标识
    app_name = models.CharField(_(u'应用名'), max_length=40, blank=True, null=True, help_text=u'应用名')
    rule_text = models.TextField(_(u'比赛规则'), max_length=255, blank=True, null=True, help_text=u'比赛规则')
    is_show_userinfo = models.BooleanField(_(u'展示用户表单'), default=False, help_text=u'是否展示用户信息表单')
    userinfo_fields = models.CharField(_(u'用户表单字段'), max_length=128, blank=True, null=True, help_text=u'需要用户填写的字段#隔开')
    userinfo_field_names = models.CharField(_('用户表单 label'), max_length=128, blank=True, null=True, help_text=u'用户需要填写的表单字段 label 名称')
    option_fields = models.CharField(_(u'下拉框字段'), max_length=128, blank=True, null=True, help_text=u'下拉框字段选项配置，#号隔开，每个字段由:h 和，号组成。 如 option1:吃饭，喝水，睡觉#option2:上班，学习，看电影')
    class Meta:
        verbose_name = _(u'应用配置信息')
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        verbose_name_plural = _(u'应用配置信息')
    def __unicode__(self):
        return str(self.pk)
    # 页面配置数据
    @property
    def show_info(self):        # 显示信息 show_info 属性
        return {
            'is_show_userinfo': self.is_show_userinfo,
            'userinfo_fields': self.userinfo_fields,
            'userinfo_field_names': self.userinfo_field_names,
            'option_fields': self.option_fields,
        }
    @property
    def text_info(self):        # 文本信息 text_info 属性
        return {
            'rule_text': self.rule_text,
        }
    @property
    def data(self):             # 总数据 data 属性
        return {
            'show_info': self.show_info,
            'text_info': self.text_info,
            'app_id': self.app_id,
            'app_name': self.app_name
        }
```

上面的模型类指定了页面需要进行的一些配置，其中，is_show_userinfo 字段用来控制展示和隐藏。具体展示成什么样？在这里将该功能做成了一张动态的表单，在 userinfo_fields 字段中保存一个字符串值，格式如下：

```
name#sex#age#phone          # 以#隔开的一个纯文本值，每一段的值代表了表单中的一个字段
```

通过上面的字符串值，当用户想要在表单中展示更多字段时，只需修改该值即可。

另外，表单中的 label 标签在 user_info_fieldnames 这个字段中给出。而如果想展示成带下拉框的情形呢？只需要在 option_fields 这个字段中定义类似下面的值即可。

```
sex:男#女,graduated_from:QingHuaDaXue,BeijingDaXue
```

在上面的代码中，每个逗号代表一个配置项，冒号用来分隔字段名和可选值。这里的 sex 是字段名，可选值是男和女两个值。

18.6.3 答题功能的实现

当单击“开始挑战”按钮时，代表已经确认过比赛信息，可以开始答题了。答题页面的效果如图 18.19 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

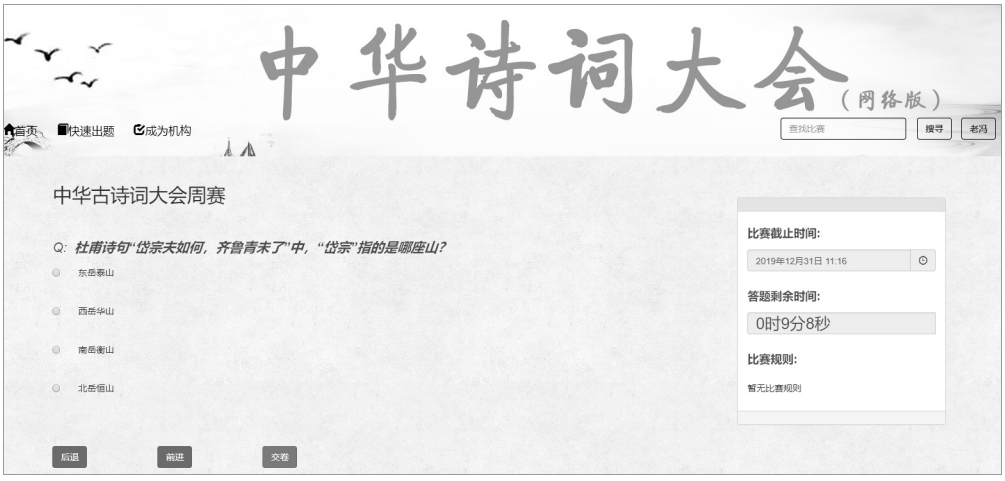


图 18.19　答题页面

添加如下的 URL 路由：

```
url(r'^game$', cop_render.game, name='game'),
```

在上面的路由中用到了 game()视图函数，该函数用来获取与比赛、题库和用户相关的信息，其详细代码如下：

```
@check_login
@check_copstatus
def game(request):
    uid = request.GET.get('uid', '')            # 获取 uid
    kind_id = request.GET.get('kind_id', '')    # 获取 kind_id
    try:  # 获取比赛信息
        kind_info = CompetitionKindInfo.objects.get(kind_id=kind_id)
    except CompetitionKindInfo.DoesNotExist:    # 未获取到渲染错误视图
        return render(request, 'err.html', CompetitionNotFound)
    try:  # 获取题库信息
        bank_info = BankInfo.objects.get(bank_id=kind_info.bank_id)
    except BankInfo.DoesNotExist:               # 未获取到渲染错误视图
        return render(request, 'err.html', BankInfoNotFound)
    try:  # 获取用户信息
        profile = Profile.objects.get(uid=uid)
    except Profile.DoesNotExist:                # 未获取到渲染错误视图
        return render(request, 'err.html', ProfileNotFound)
    if kind_info.question_num > bank_info.total_question_num:  # 检查题库大小
        return render(request, 'err.html', QuestionNotSufficient)
    pageconfig = get_pageconfig(kind_info.app_id)              # 获取页面配置信息
    return render(request, 'competition/game.html', {          # 渲染视图信息
        'user_info': profile.data,
        'kind_id': kind_info.kind_id,
        'kind_name': kind_info.kind_name,
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        'cop_finishat': kind_info.cop_finishat,
        'period_time': kind_info.period_time,
        'rule_text': pageconfig.get('text_info', {}).get('rule_text', '')
    })
```

当加载比赛页面的时候，只是获取到了基本数据，对于题目信息，需要使用 AJAX 异步请求的方式进行获取，代码如下：

```
var currentPage = 1;
var hasPrevious = false;
var hasNext = false;
var questionNum = 0;
var response;
var answerDict;
  $(document).ready(function () {
      if({{ period_time|safe }}) {
          startTimer1();
      }
      $('#loadingModal').modal('show');
      uid = '{{ user_info.uid|safe }}';
      kind_id = '{{ kind_id|safe }}';
      $.ajax({
          url: '/api/questions',
          type: 'get',
          data: {
              'uid': uid,
              'kind_id': kind_id
          },
          dataType: 'json',
          success: function (res) {
              response = res;
              questionNum = res.data.kind_info.question_num;
              answerDict = new Array(questionNum);
              for(var i=0; i < questionNum; i++){
                  if(response.data.questions[i].qtype === 'choice') {
                      answerDict['c_' + response.data.questions[i].pk] = '';
                  }else{
                      answerDict['f_' + response.data.questions[i].pk] = '';
                  }
              }
              if(res.data.questions[0].qtype === 'choice') {
                  $('#question').html(res.data.questions[0].question); // currentPage - 1
                  $('#item1').html(res.data.questions[0].items[0]);
                  $('#item2').html(res.data.questions[0].items[1]);
                  $('#item3').html(res.data.questions[0].items[2]);
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
                $('#item4').html(res.data.questions[0].items[3]);
                $('#itemPk').html('c_' + res.data.questions[0].pk);
                hasNext = (currentPage < questionNum);
                $('#fullinBox').hide();
            } else{
                $('#question').html(res.data.questions[0].question.replace('##',
'_____'));
                $('#answerPk').val('f_' + res.data.questions[0].pk);
                hasNext = (currentPage < questionNum);
                $('#choiceBox').hide();
            }
            $('#loadingModal').modal('hide');
        }
    });
```

由于需要从题库中随机抽取指定数目的题目，所以在/api/questions 目录中的 competition app 下面添加一个接口视图 game_views.py。视图代码如下：

```
import datetime                                          # datetime 模块
import random                                            # random 模块
from django.db import transaction                        # 数据库事务模块
from django.views.decorators.csrf import csrf_exempt  # 防跨站请求伪造修饰器
from account.models import Profile, UserInfo             # Profile 和 UserInfo 模型类
from competition.models import (BankInfo, ChoiceInfo, CompetitionKindInfo,
                          CompetitionQAInfo, FillInBlankInfo) # 比赛模型类
from TimeConvert import TimeConvert as tc                # 第三方时间转换工具
from utils.check_utils import check_correct_num   # 检查答题是否正确的小工具
from utils.decorators import check_copstatus, check_login  # 修饰器
from utils.errors import BankInfoNotFound, CompetitionError, ProfileError  # 错误码
from utils.redis.rprofile import enter_userinfo   # 用户填写的表单信息 Redis 值
from utils.redis.rrank import add_to_rank          # 将排行榜存入 Redis 方法中
from utils.response import json_response          # 响应类
@check_login
@check_copstatus
@transaction.atomic
def get_questions(request):
    kind_id = request.GET.get('kind_id', '')       # 获取 kind_id
    uid = request.GET.get('uid', '')               # 获取 uid
    try:  # 获取比赛信息
        kind_info = CompetitionKindInfo.objects.select_for_update().get(kind_id=
kind_id)
    except CompetitionKindInfo.DoesNotExist:       # 未获取到，返回错误码 100001
        return json_response(*CompetitionError.CompetitionNotFound)
    try:                                           # 获取题库信息
        bank_info = BankInfo.objects.get(bank_id=kind_info.bank_id)
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    except BankInfo.DoesNotExist:                    # 未获取到，返回错误码 100004
        return json_response(*CompetitionError.BankInfoNotFound)
    try:                                             # 获取用户信息
        profile = Profile.objects.get(uid=uid)
    except Profile.DoesNotExist:                     # 未获取到，返回错误码 200001
        return json_response(*ProfileError.ProfileNotFound)
    qc = ChoiceInfo.objects.filter(bank_id=kind_info.bank_id)      # 选择题
    qf = FillInBlankInfo.objects.filter(bank_id=kind_info.bank_id) # 填空题
    questions = []                                   # 将两种题型放到同一个列表中
    for i in qc.iterator():
        questions.append(i.data)
    for i in qf.iterator():
        questions.append(i.data)
    question_num = kind_info.question_num            # 出题数
    q_count = bank_info.total_question_num           # 总题数
    if q_count < question_num:                # 出题数大于总题数，返回错误码 100005
        return json_response(CompetitionError.QuestionNotSufficient)
    qs = random.sample(questions, question_num)      # 随机分配题目
    qa_info = CompetitionQAInfo.objects.select_for_update().create(  # 创建答题 log 数据
        kind_id=kind_id,
        uid=uid,
        qsrecord=[q['question'] for q in qs],
        asrecord=[q['answer'] for q in qs],
        total_num=question_num,
        started_stamp=tc.utc_timestamp(ms=True, milli=True),  # 设置开始时间戳
        started=True
    )
    for i in qs:  # 剔除答案信息
        i.pop('answer')
    return json_response(200, 'OK', {  # 返回 JSON 格式的数据，包括题目信息、答题 log 信息等
        'kind_info': kind_info.data,
        'user_info': profile.data,
        'qa_id': qa_info.qa_id,
        'questions': qs
    })
```

上面的 api 视图需要在 api 模块下的 urls.py 文件中配置路由，代码如下：

```
url(r'^questions$', game_views.get_questions, name='get_questions'),
```

这个接口主要用于生成比赛数据。比赛数据是从题库中随机抽取的指定数目的题目。每次调用接口，都会返回不同的结果。每次调用接口都会生成一个答题日志。对于没有答题就刷新了页面的用户，日志不会丢失，而会被标记为未完成。

注意答题是有时间限制的，该时间限制在 CompetitionKindInfo 数据模型的 period_time 字段中配置。如果用户答题超过了这个时间，则答题日志会被标记为已超时，并且答题数据会存

電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

在，供以后数据分析用，但是不会参与到排行榜中，这在答题中会有相应的提示。

答题数据统一返回到页面中，页面需要按照页面大小为 1 的数量对返回的题目进行分页，并且需要记住用户上一道题和下一道题的答题情况和顺序。这些功能需要在前台实现，可以参考资源包中的源代码。

18.6.4　提交答案

在答题完成后，需要判断答题剩余时间，如果剩余时间为 0，或者已经超时，则把答题的日志保存为超时，并且答题成绩不能加入排行榜；而如果剩余时间还很充足，那么用户的成绩要加入排行榜，并且将答题日志标记为已完成，用来区别未完成的答题记录。提交答案显示成绩单页面的效果如图 18.20 所示。

图 18.20　提交答案显示成绩单页面

在答题过程中，前端需要记录用户的答题情况和顺序，并生成一个指定的数据形式，以便提交到后台进行答案的匹配。提交答案的实现代码如下：

```
$('#answerSubmit').click(function () {
      if(window.confirm("确认提交答案吗?")) {
          if({{ period_time|safe }}) {
              stopTimer1();
          }
          var answer = "";
          for (var key in answerDict) {
              if (!answer) {
                  answer = String(key) + "," + answerDict[key] + "#";
              }else{
                  answer += String(key) + "," + answerDict[key] + "#";
              }
          }
          $.ajax({
              url: '/api/answer',
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            type: 'post',
            data: {
                'qa_id': response.data.qa_id,
                'uid': response.data.user_info.uid,
                'kind_id': kind_id,
                'answer': answer
            },
            dataType: 'json',
            success: function (res) {
                if(res.status === 200) {
                    window.location.href = "/bs/result?uid=" + res.data.user_
info.uid + "&kind_id=" + res.data.kind_id + "&qa_id=" + res.data.qa_id;
                }else{
                    alert('提交失败');
                }
            }
        })
    }else {}
  })
});
```

/api/answer 接口对应的路由要在 api 模块下的 urls.py 文件中填写，代码如下：

```
url(r'^answer$', game_views.submit_answer, name='submit_answer'),
```

上面用到了 submit_answer()视图函数，该视图函数需要添加到 game_views.py 文件中，实现代码如下：

```
@csrf_exempt
@check_login
@check_copstatus
@transaction.atomic
def submit_answer(request):
    stop_stamp = tc.utc_timestamp(ms=True, milli=True) # 结束时间戳
    qa_id = request.POST.get('qa_id', '')              # 获取 qa_id
    uid = request.POST.get('uid', '')                  # 获取 uid
    kind_id = request.POST.get('kind_id', '')          # 获取 kind_id
    answer = request.POST.get('answer', '')            # 获取 answer
    try:  # 获取比赛信息
        kind_info = CompetitionKindInfo.objects.get(kind_id=kind_id)
    except CompetitionKindInfo.DoesNotExist:           # 未获取到,返回错误码 100001
        return json_response(*CompetitionError.CompetitionNotFound)
    try:  # 获取题库信息
        bank_info = BankInfo.objects.get(bank_id=kind_info.bank_id)
    except BankInfo.DoesNotExist:                      # 未获取到,返回错误码 100004
        return json_response(*CompetitionError.BankInfoNotFound)
    try:  # 获取用户信息
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        profile = Profile.objects.get(uid=uid)
    except Profile.DoesNotExist:                          # 未获取到,返回错误码 200001
        return json_response(*ProfileError.ProfileNotFound)
    try:  # 获取答题 log 信息
        qa_info = CompetitionQAInfo.objects.select_for_update().get(qa_id=qa_id)
    except CompetitionQAInfo.DoesNotExist:                # 未获取到,返回错误码 100006
        return json_response(*CompetitionError.QuestionNotFound)

    answer = answer.rstrip('#').split('#')                # 处理答案数据
    total, correct, wrong = check_correct_num(answer)  # 检查答题情况
    qa_info.aslogrecord = answer
    qa_info.finished_stamp = stop_stamp
    qa_info.expend_time = stop_stamp - qa_info.started_stamp
    qa_info.finished = True
    qa_info.correct_num = correct if total == qa_info.total_num else 0
    qa_info.incorrect_num = wrong if total == qa_info.total_num else qa_info.
total_num
    qa_info.save()                                        # 保存答题 log
    if qa_info.correct_num == kind_info.question_num:  # 得分处理
        score = kind_info.total_score
    elif not qa_info.correct_num:
        score = 0
    else:
        score = round((kind_info.total_score / kind_info.question_num) * correct, 3)
    qa_info.score = score                                 # 继续保存答题 log
    qa_info.save()
    kind_info.total_partin_num += 1                       # 保存比赛数据
    kind_info.save()                                      # 比赛答题次数
    bank_info.partin_num += 1
    bank_info.save()                                      # 题库答题次数
    if (kind_info.period_time > 0) and (qa_info.expend_time > kind_info.
period_time * 60 * 1000):                                 # 超时，不加入排行榜
        qa_info.status = CompetitionQAInfo.OVERTIME
        qa_info.save()
    else:                                                 # 正常完成，加入排行榜
        add_to_rank(uid, kind_id, qa_info.score, qa_info.expend_time)
        qa_info.status = CompetitionQAInfo.COMPLETED
        qa_info.save()
    return json_response(200, 'OK', {                     # 返回 JSON 格式的数据
        'qa_id': qa_id,
        'user_info': profile.data,
        'kind_id': kind_id,
    })
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

18.6.5 批量录入题库

录入题库功能的实现方法是：在页面中为用户提供一个 Excel 模板，用户按照对应的模板格式来编写题库信息，编写完成后，在页面中选择带有题库的 Excel 文件，单击“开始录制”按钮，进行题库的录入。题库 Excel 模板如图 18.21 所示。题库录入界面如图 18.22 所示。

	A	B	C	D	E	F
1	问题	答案	选项一	选项二	选项三	选项四
29	杜甫诗句“岱宗夫如何，齐鲁青未了”中，“岱宗”指的是哪座山？	东岳泰山	东岳泰山	西岳华山	南岳衡山	北岳恒山
30	下列哪个词牌和春秋战国时期的弄玉公主有关？	忆秦娥	浪淘沙	贺新郎	忆秦娥	念奴娇
31	《三国演义》的开篇词《临江仙》的作者是谁？	杨慎	罗贯中	杨慎	施耐庵	曹雪芹
32	##，八千里路云和月	三十功名尘与土				
33	《山坡羊 潼关怀古》的作者张养浩是哪个朝代的人？	元朝	宋朝	元朝	明朝	清朝
34	《岳阳楼记》是哪位朋友拜托范仲淹创作的？	滕子京	滕子京	王安石	欧阳修	司马光
35	“横看成岭侧成峰，远近高低各不同”说的是哪座山的风光？	庐山	天门山	黄山	终南山	庐山
36	杜甫诗句“群山万壑赴荆门，生长明妃尚有村”中，“明妃”指的是谁？	王昭君	李清照	薛涛	王昭君	赵飞燕
37	《孔雀东南飞》中的女主人公刘兰芝在多大年龄结婚的？	十七岁	十三岁	十五岁	十七岁	二十岁
38	辛弃疾写过《南乡子 登京口北固亭有怀》，其中作者最佩服的人是谁？	孙权	曹操	刘备	孙权	周瑜
39	陶渊明在《归园田居》中记叙自己做官多少年？	三十年	十年	二十年	三十年	四十年
40	唐伯虎诗句“头上红冠不用裁，满身雪白走将来。平生不敢轻言语，一叫千门万户开。”描述的是那种动物？	鸡	鸡	鸭	鹅	燕
41	人生自古谁无死，##	留取丹心照汗青				
42	《霸王别姬》中的女主人公虞姬与哪个词牌有关？	虞美人	西江月	虞美人	沁园春	雨霖铃
43	苏轼诗句“东风袅袅泛崇光，香雾空蒙月转廊。只恐夜深花睡去，故烧高烛照红妆。”描述的是哪种花？	海棠花	杏花	梨花	梅花	海棠花
44	《观沧海》是曹操在哪座关附近创作的？	山海关	函谷关	山海关	虎牢关	玉门关
45	“杨花落尽子规啼，闻道龙标过五溪”是李白为哪位朋友写的诗句？	王昌龄	王昌龄	王之涣	孟浩然	杜甫
46	说明：请按照表格对应的行和列填写题目。					
47						
48	对于题型为选择题的问题，题目填写到第一列，答案填写到第二列，四个选项分别填写到第三到第六列。如果题目中有图片需要展示，请在第七列填写图片的链接，如果题目中存在音频，请在第八列填写音频链接。					
49						
50	对于题型为填空题的问题，题目填写到第一列，答案填写到第二列。					
51	注意：问题中需要补全答案的位置，不论要补全的字数有多少，都用两个井号代替 ##，如： 慈母手中线，##身上衣。另外，填空题不存在选项，所以第三列到第六列不用填写。					
52	如果题目中有图片需要展示，请在第七列填写图片的链接，如果题目中存在音频，请在第八列填写音频链接。					
53						
54						

图 18.21　题库 Excel 模板

图 18.22　题库录入界面

综上所述，录入题库主要分为以下 5 个步骤。

（1）下载 Excel 模板文件。

（2）根据自己的题库需求修改 Excel 模板文件。

（3）输入题库名称并选择题库类型。

（4）上传文件。

（5）提交到数据库。

下面详细讲解录入题库功能的实现过程。在前端给配置题库添加一个导航页，在 competition app 下面的 urls.py 文件中添加如下几条路由：

```
# 配置比赛 URL
urlpatterns += [
```


電子工業出版社 PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    url(r'^set$', set_render.index, name='set_index'),          # 配置导航页
    url(r'^set/bank$', set_render.set_bank, name='set_bank'),  # 配置题库页面
    url(r'^set/bank/tdownload$', set_render.template_download, name='template_
download'),         # 题库模板下载
    # 上传题库
    url(r'^set/bank/upbank$', set_render.upload_bank, name='upload_bank'),
    url(r'^set/game$', set_render.set_game, name='set_game'),  # 配置比赛页
]
```

在 competition app 下面添加一个 render 视图模块 set_render.py，并在其中添加 index()函数，用来渲染视图和用户信息数据。代码如下：

```
import os
from django.conf import settings                                # 配置文件
from django.db import transaction                               # 数据库事务
from django.http.response import StreamingHttpResponse          # 流式响应对象
from django.shortcuts import render                             # render 函数
from account.models import Profile, UserInfo                    # 两个用户信息模型类
from competition.models import BankInfo                         # 题库信息模型类
from business.models import BusinessAccountInfo                 # 机构账户信息模型类
from utils.decorators import check_login                        # 登录检查修饰器
from utils.errors import (FileNotFound, FileTypeError, ProfileNotFound,  # 错误信息
                          TemplateNotFound, BizAccountNotFound)
from utils.small_utils import get_now_string, get_today_string        # 工具
from utils.upload_questions import upload_questions             # 上传题库函数
@check_login
def index(request):
    """
    题库和比赛导航页
    :param request: 请求对象
    :return: 渲染视图和 user_info 用户信息数据
    """
    uid = request.GET.get('uid', '')
    try:
        profile = Profile.objects.get(uid=uid)
    except Profile.DoesNotExist:
        return render(request, 'err.html', ProfileNotFound)
    return render(request, 'setgames/index.html', {'user_info': profile.data})
```

导航页使用了 Bootstrap 框架的巨幕 jumbotron。当用户单击“录制题库”按钮时，页面跳转到 urls.py 中的第二条路由，对应的视图函数代码如下：

```
@check_login
def set_bank(request):
    """
    配置题库页面
    :param request: 请求对象
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    :return: 渲染页面返回 user_info 用户信息数据和 bank_types 题库类型数据
    """
    uid = request.GET.get('uid', '')
    try:
        profile = Profile.objects.get(uid=uid)      # 检查账户信息
    except Profile.DoesNotExist:
        return render(request, 'err.html', ProfileNotFound)
    bank_types = []
    for i, j in BankInfo.BANK_TYPES:                # 返回所有题库类型
        bank_types.append({'id': i, 'name': j})
    return render(request, 'setgames/bank.html', { # 渲染模板
        'user_info': profile.data,
        'bank_types': bank_types
    })
```

对应的 HTML 模板放置在 web/setgames/bank.html 中，关键代码如下：

```
<form id="uploadFileForm" method="post" action="/bs/set/bank/upbank" enctype=
"multipart/form-data">{% csrf_token %}
<div id="uploadMainRow" class="row" style="margin-top: 120px;">
    <div class="col-md-3">
        <label>① 下载题库</label>
        <p style="color: gray;margin-top: 5px;"><a id="tDownload" href="/bs/set/
bank/tdownload?uid={{ user_info.uid }}" class="btn     btn-success btn-sm">下载
</a>我们的简易模板，按照模板中的要求修改题库。</p>
    </div>
    <div class="col-md-3">
        <div class="form-group">
            <label for="bankName">② 题库名称</label>
            <input id="bankName" name="bank_name" type="text" class="form-control"
placeholder="请输入题库名称" />
        </div>
    </div>
    <div class="col-md-3">
        <label for="choicedValue">③ 题库类型</label>
        <div class="dropdown">
            <input type="button" id="choicedValue" class="btn btn-default dropdown-
toggle form-control" data-toggle="dropdown" name="bank_type" value="选择一个题库
类型" />
            <div class="dropdown-menu">
                {% for t in bank_types %}
                    <div class="btn btn-default form-control" onclick="choiceBankType
(this)">{{ t.name }}</div>
                {% endfor %}
            </div>
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        </div>
    </div>
    <div class="col-md-3">
        <div class="row" style="margin-left:-1px;">
            <label for="uploadFile">④ 上传文件</label>
            <input class="form-control" name="template" type="file" style="text-
align: inherit" id="uploadFile">
        </div>
    </div>
    <input type="hidden" name="uid" value="{{ user_info.uid }}" />
</div>
<div class="row" style="margin-top:35px;">
    <input type="submit" id="startUpload" class="btn btn-danger" style="float:
right" value="开始录制">
</div>
</form>
<script type="text/javascript">
    var choicedBankType;
    var responseTypes = {{ bank_types|safe }};
    var choiceBankType = function (t) {
        var cbt = $(t).html();
        for(var i in responseTypes){
            if(responseTypes[i].name === cbt){
                choicedBankType = responseTypes[i].id;
                break;
            }
        }
        $('#choicedValue').val(cbt);
    }

</script>
```

在开始录入题库之前，用户需要先下载 Excel 模板文件进行编辑后才能提交。下载的 URL 路由在 urls.py 中的第三行，对应的视图函数代码如下：

```
@check_login
def template_download(request):
    uid = request.GET.get('uid', '')                # 获取 uid
    try:
        Profile.objects.get(uid=uid)                # 用户信息
    except Profile.DoesNotExist:
        return render(request, 'err.html', ProfileNotFound)
    def iterator(file_name, chunk_size=512):        # chunk_size 大小为 512KB
        with open(file_name, 'rb') as f:            # rb，以字节读取
            while True:
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            c = f.read(chunk_size)
            if c:
                yield c  # 使用 yield 返回数据，直到所有数据返回完毕才退出
            else:
                break
    template_path = 'web/static/template/template.xlsx'
    # 希望保存题库文件到一个单独目录
    file_path = os.path.join(settings.BASE_DIR, template_path)
    if not os.path.exists(file_path):                # 路径不存在
        return render(request, 'err.html', TemplateNotFound)
    # 将文件以流式响应返回到客户端
    response = StreamingHttpResponse(iterator(file_path), content_type=
'application/vnd.ms-excel')
    # 格式为.xlsx
    response['Content-Disposition'] = 'attachment; filename=template.xlsx'
    return response
```

当用户单击“开始录制”按钮时，数据以 POST 方式提交到后台，该视图函数对应的 URL 在 urls.py 中的第四行，视图函数代码如下：

```
@check_login
@transaction.atomic
def upload_bank(request):
    uid = request.POST.get('uid', '')                # 获取 uid
    bank_name = request.POST.get('bank_name', '')  # 获取题库名称
    bank_type = int(request.POST.get('bank_type', BankInfo.CATE0)) # 获取题库类型
    template = request.FILES.get('template', None) # 获取模板文件
    if not template:                                 # 模板文件不存在
        return render(request, 'err.html', FileNotFound)
    if template.name.split('.')[-1] not in ['xls', 'xlsx']: # 模板文件格式为.xls 或.xlsx
        return render(request, 'err.html', FileTypeError)
try:  # 获取用户信息
        profile = Profile.objects.get(uid=uid)
    except Profile.DoesNotExist:
        return render(request, 'err.html', ProfileNotFound)
bank_info = BankInfo.objects.select_for_update().create(  # 创建题库 BankInfo
        uid=uid,
        bank_name=bank_name or '暂无',
        bank_type=bank_type
    )
    # 保存文件目录以当天时间为准
    today_bank_repo = os.path.join(settings.BANK_REPO, get_today_string())
    if not os.path.exists(today_bank_repo):
        os.mkdir(today_bank_repo)                    # 不存在该目录则创建
    final_path = os.path.join(today_bank_repo, get_now_string(bank_info.bank_id)) +
'.xlsx'  # 生成文件名
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    with open(final_path, 'wb+') as f:              # 保存到目录
        f.write(template.read())
    # 使用 xlrd 读取 Excel 文件到数据库
    choice_num, fillinblank_num = upload_questions(final_path, bank_info)
    return render(request, 'setgames/bank.html', { # 渲染视图
        'user_info': profile.data,
        'created': {
            'choice_num': choice_num,
            'fillinblank_num': fillinblank_num
        }
    })
```

上面的视图函数首先将返回的 Excel 题库模板保存到指定目录，以便于后期使用；然后生成一个题库 BankInfo，并使用一个自定义的 Python 脚本将 Excel 题库文件中的数据逐一读取出来，保存到数据库中。

上面视图对应的函数文件放置在 utils 模块下的 upload_questions.py 文件中，代码如下：

```
import xlrd                          # xlrd 库
from django.db import transaction  # 数据库事务
from competition.models import ChoiceInfo, FillInBlankInfo  # 题目数据模型
def check_vals(val):                 # 检查值是否被转换成 float。如果是，则将.0 结尾去掉
    val = str(val)
    if val.endswith('.0'):
        val = val[:-2]
    return val
@transaction.atomic
def upload_questions(file_path=None, bank_info=None):
    book = xlrd.open_workbook(file_path)    # 读取文件
    table = book.sheets()[0]                # 获取第一张表
    nrows = table.nrows                     # 获取行数
    choice_num = 0                          # 选择题数量
    fillinblank_num = 0                     # 填空题数量
    for i in range(1, nrows):
        rvalues = table.row_values(i)       # 获取行中的值
        if (not rvalues[0]) or rvalues[0].startswith('说明'):  # 取出多余行
            break
        if '##' in rvalues[0]:              # 选择题
            FillInBlankInfo.objects.select_for_update().create(
                bank_id=bank_info.bank_id,
                question=check_vals(rvalues[0]),
                answer=check_vals(rvalues[1]),
                image_url=rvalues[6],
                source=rvalues[7]
            )
            fillinblank_num += 1  # 填空题数加 1
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        else:  # 填空题
            ChoiceInfo.objects.select_for_update().create(
                bank_id=bank_info.bank_id,
                question=check_vals(rvalues[0]),
                answer=check_vals(rvalues[1]),
                item1=check_vals(rvalues[2]),
                item2=check_vals(rvalues[3]),
                item3=check_vals(rvalues[4]),
                item4=check_vals(rvalues[5]),
                image_url=rvalues[6],
                source=rvalues[7]
            )
            choice_num += 1  # 选择题数加1
    bank_info.choice_num = choice_num
    bank_info.fillinblank_num = fillinblank_num
    bank_info.save()
    return choice_num, fillinblank_num
```

录入题库的过程也非常简单，只需使用 xlrd 读取文件中的每一行，判断第一列的题目信息中是否包含##，如果包含，就代表该题目是填空题。在答题的时候，页面会将##解读为 4 条下画线（____），方便用户答题。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 6 篇　商业项目篇

第 19 章

图像识别工具

19.1　项目概述

在一张图片中获取图片上的相关信息是一件很麻烦的事情。本章将通过 Python+PyQt5+百度 AI 接口实现图像识别工具项目。本项目可以识别银行卡图片、植物图片、动物图片、通用票据图片、营业执照图片、身份证图片、车牌号图片、驾驶证图片、行驶证图片、车型图片、Logo 图片等图片中的相关信息。

19.2　设计流程

图像识别工具的设计流程如图 19.1 所示。

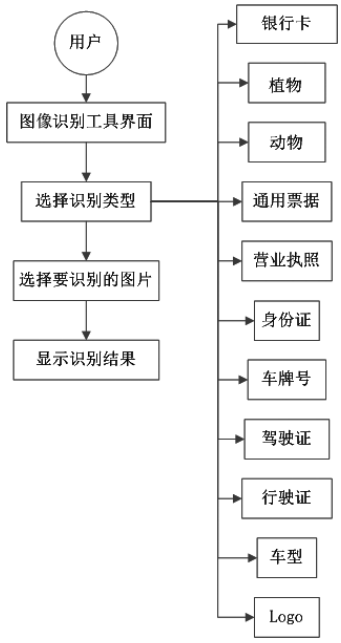


图 19.1　图像识别工具的设计流程

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

19.3 系统预览

图像识别工具默认显示效果如图 19.2 所示。

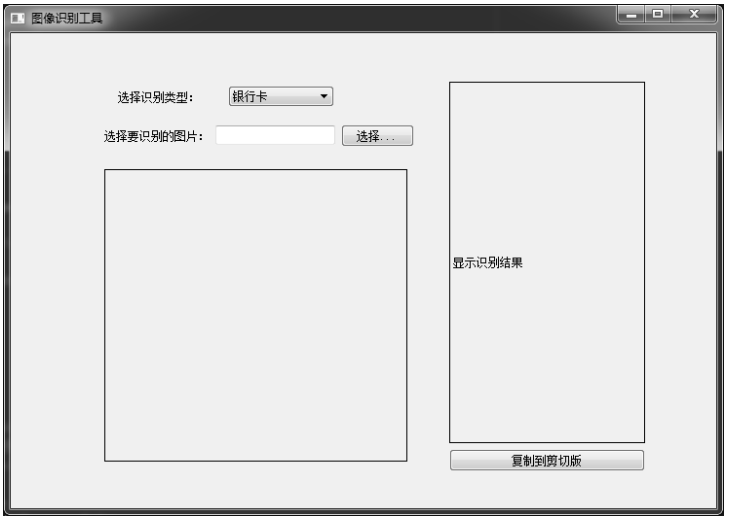


图 19.2　图像识别工具默认显示效果

选择识别类型，单击默认显示银行卡的下拉按钮，如图 19.3 所示。

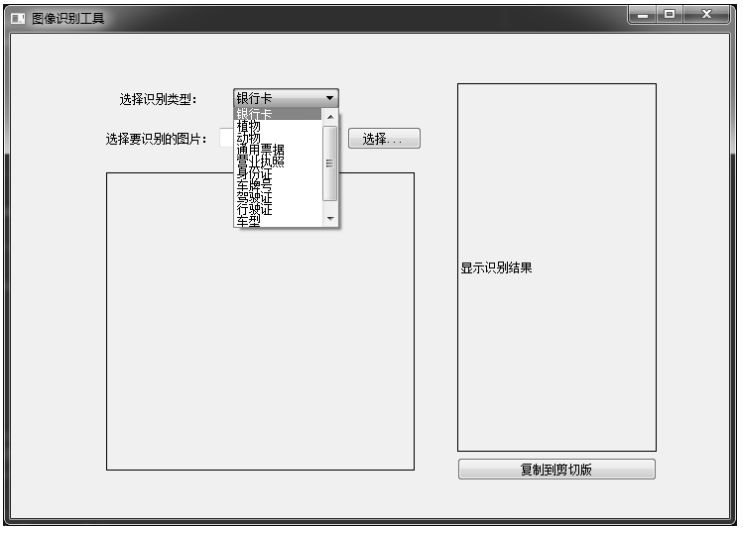


图 19.3　选择识别类型

选择完类别后，单击“选择”按钮，弹出“选择要识别的图片”对话框，如图 19.4 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 19.4　“选择要识别的图片”对话框

找到想要识别的图片，选择完成后，将会根据选择的类型，显示图片中的相关信息。以驾驶证图片识别为例，识别结果如图 19.5 所示。

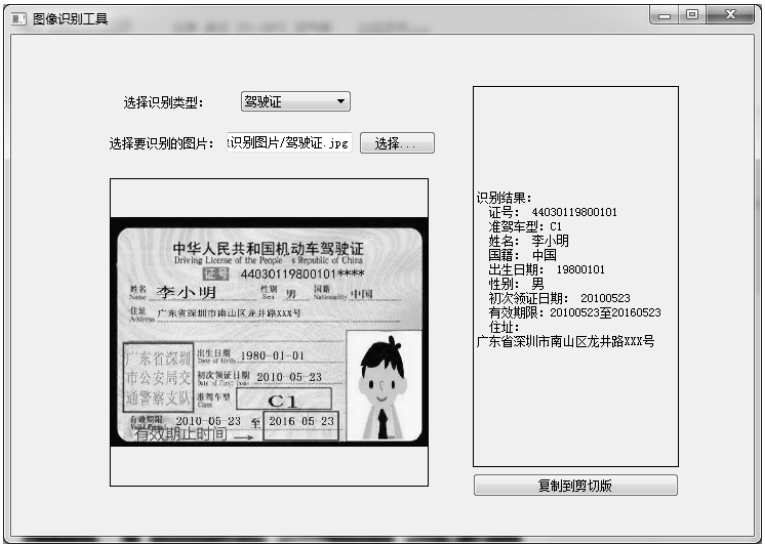


图 19.5　显示驾驶证图片识别结果

19.4　开发工具准备

- 操作系统：Windows 7、Windows 8、Windows 10 或 Ubuntu。
- 开发工具：PyCharm、Qt。
- 开发模块：urllib、urllib.request、Base64、JSON、PyQt5。
- API 接口：百度 API 接口。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

19.5 开发前的准备工作

在开发图像识别工具项目时，主要使用了 PyQt5 搭建界面，以及 Base64、urllib、urllib.request 这 3 种模块来获取百度 API 接口信息，然后使用 JSON 模块解析返回的 JSON 格式的数据。

19.5.1 申请百度 AI 接口

申请地址：http://ai.baidu.com/。

图像识别主要使用的就是在百度 AI 开放平台申请的接口。访问上面的申请地址后，单击“控制台”，再单击“图像识别”，如图 19.6 所示。

图 19.6 单击“图像识别”

单击“图像识别”后，会提示进入登录页面，如图 19.7 所示。

图 19.7 登录页面

登录成功后，进入控制台，依次单击“产品服务”→“全部产品”→“图像识别”，如图 19.8 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 19.8　在控制台中单击“图像识别”

进入“图像识别”页面后，选择“创建应用”，添加应用名称，然后根据项目需求勾选多个接口权限（默认仅有图像识别权限），最后单击“立即创建”按钮，完成应用的创建。应用创建完成后，进入“应用列表”页面，在该页面中查看项目中需要使用的 API Key、Secret Key 的值，如图 19.9 所示。

图 19.9　查询应用 API Key、Secret Key

19.5.2　urllib、urllib.request 模块

urllib 是 Python 内置的 HTTP 请求库，是 Python 的一个获取 URL，可以用来抓取远程的数据。urllib.request 为请求块。

导入 urllib 库，然后使用 urllib.request 请求块，代码如下：

```
01 import urllib.request
02
03 # client_id 为官网获取的 AK，client_secret 为官网获取的 SK
04 host = 'https://aip.baidubce.com/oauth/2.0/token?grant_type=client_credentials&
client_id=' + API_KEY + '&client_secret=' + SECRET_KEY
05 # 发送请求
06 request = urllib.request.Request(host)
07 # 添加请求头
08 request.add_header('Content-Type', 'application/json; charset=UTF-8')
09 # 获取返回内容
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
response = urllib.request.urlopen(request)
# 读取返回内容
content = response.read()
```

> **说明：**以上网络请求的目标地址为百度 AI 图像识别工具的接口，详细信息可以参考官网中的 API 文档。

19.5.3 JSON 模块

JSON 是一种轻量级的数据交换格式。Python 3 可以使用 JSON 模块来对 JSON 格式的数据进行编/解码。

1．JSON 模块的常用方法

- json.dump()：将 Python 数据对象以 JSON 格式数据流的形式写入文件中。
- json.load()：解析包含 JSON 格式数据的文件为 Python 对象。
- json.dumps()：将 Python 数据对象转换为 JSON 格式的字符串。
- json.loads()：将 JSON 格式的字符串、字节及字节数组解析为 Python 对象。

2．JSON 模块的使用

导入 JSON 库，然后使用 JSON 解析 JSON 格式的数据，代码如下：

```
#!/usr/bin/python3
import json
# 将 Python 字典类型转换为 JSON 对象
data1 = {
    'no' : 1,
    'name' : 'Runoob',
    'url' : 'http://www.runoob.com'
}
json_str = json.dumps(data1)
print ("Python 原始数据：", repr(data1))
print ("JSON 对象：", json_str)
# 将 JSON 对象转换为 Python 字典类型
data2 = json.loads(json_str)
print ("data2['name']: ", data2['name'])
print ("data2['url']: ", data2['url'])
```

执行以上代码，输出结果为：

```
Python 原始数据:  {'name': 'Runoob', 'no': 1, 'url': 'http://www.runoob.com'}
JSON 对象:  {"name": "Runoob", "no": 1, "url": "http://www.runoob.com"}
data2['name']:  Runoob
data2['url']:  http://www.runoob.com
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

19.6　图像识别工具的开发

19.6.1　根据原型设计制作窗体

在设计图像识别工具的主窗体时，首先需要创建主窗体外层，然后依次添加分类选择部分、图片选择部分、选择的图片显示部分、识别结果显示部分、复制识别结果部分。窗体设计效果如图 19.10 所示。

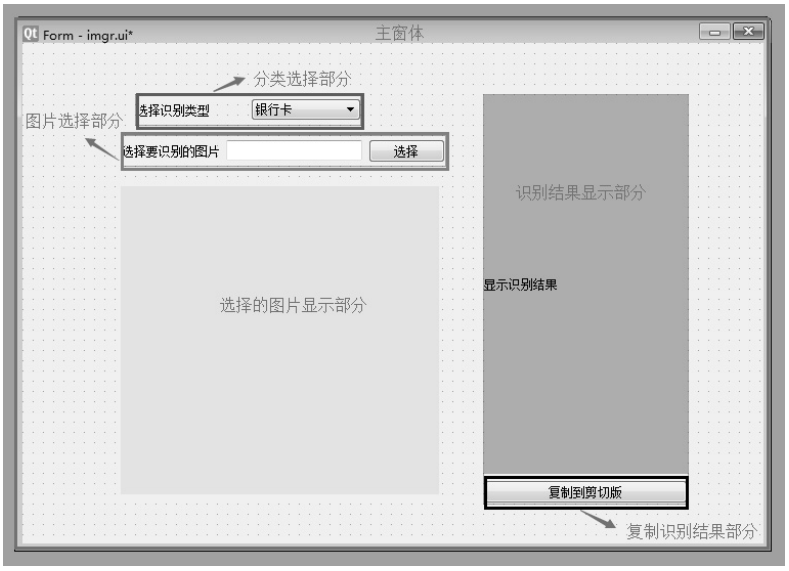


图 19.10　窗体设计效果

19.6.2　添加分类

根据原型分析，有银行卡、植物、动物、通用票据、营业执照、身份证、车牌号、驾驶证、行驶证、车型、Logo 等分类。需要添加分类到控件 QComoBox 中，代码如下：

```
# 设置下拉控件选项内容
self.comboBox.setItemText(0, _translate("Form", "银行卡"))
self.comboBox.setItemText(1, _translate("Form", "植物"))
self.comboBox.setItemText(2, _translate("Form", "动物"))
self.comboBox.setItemText(3, _translate("Form", "通用票据"))
self.comboBox.setItemText(4, _translate("Form", "营业执照"))
self.comboBox.setItemText(5, _translate("Form", "身份证"))
self.comboBox.setItemText(6, _translate("Form", "车牌号"))
self.comboBox.setItemText(7, _translate("Form", "驾驶证"))
self.comboBox.setItemText(8, _translate("Form", "行驶证"))
self.comboBox.setItemText(9, _translate("Form", "车型"))
self.comboBox.setItemText(10, _translate("Form", "Logo"))
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

19.6.3 选择要识别的图片

选择要识别图片的功能描述：单击按钮后弹出选择框，进行图片选择，选择图片后显示图片路径及图片，同时根据选择的分类去进行图片的识别。

（1）为按钮添加单击事件。代码如下：

```
# 为按钮设置方法
self.pushButton.clicked.connect(self.openfile)
```

（2）实现新建 openfile 按钮单击事件方法，在该方法中打开“文件选择”对话框查找图片，返回选择的图片，进行相应的处理，包括显示图片、设置显示图片路径、调用创建的 typeTp() 方法判断选择类型进行图片识别。代码如下：

```
# 打开“文件选择”对话框方法
def openfile(self):
    # 启动“文件选择”对话框，查找.jpg 及.png 图片
    self.download_path = QFileDialog.getOpenFileName(self.widget1, "选择要识别的图片", "/", "Image Files(*.jpg *.png)")
    # 判断是否选择图片
    if not self.download_path[0].strip():
        # 没有选择图片
        pass
    else:
        # 选择图片执行以下内容
        print(self.download_path[0])
        # 设置图片路径
        self.lineEdit.setText(self.download_path[0])
        # 利用 pixmap 解析图片
        pixmap = QPixmap(self.download_path[0])
        # 等比例缩放图片
        scaredPixmap = pixmap.scaled(QSize(311, 301), aspectRatioMode=Qt.KeepAspectRatio)
        # 设置图片
        self.image.setPixmap(scaredPixmap)
        # 判断选择的类型，根据类型做相应的图片处理
        self.image.show()
        # 判断选择的类型
        self.typeTp()
        pass
```

（3）实现分类方法 typeTp()，判断选择的分类进行图片识别。代码如下：

```
# 判断选择的类型进行相应处理
def typeTp(self):
    # 银行卡识别
    if self.comboBox.currentIndex() == 0:
        self.get_bankcard(self.get_token())
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        pass
    # 植物识别
    elif self.comboBox.currentIndex() == 1:
        self.get_plant(self.get_token())
        pass
    # 动物识别
    elif self.comboBox.currentIndex() == 2:
        self.get_animal(self.get_token())
        pass
    #通用票据识别
    elif self.comboBox.currentIndex() == 3:
        self.get_vat_invoice(self.get_token())
        pass
    # 营业执照识别
    elif self.comboBox.currentIndex() == 4:
        self.get_business_licensev(self.get_token())
        pass
    # 身份证识别
    elif self.comboBox.currentIndex() == 5:
        self.get_idcard(self.get_token())
        pass
    # 车牌号识别
    elif self.comboBox.currentIndex() == 6:
        self.get_license_plate(self.get_token())
        pass
    # 驾驶证识别
    elif self.comboBox.currentIndex() == 7:
        self.get_driving_license(self.get_token())
        pass
    # 行驶证识别
    elif self.comboBox.currentIndex() == 8:
        self.get_vehicle_license(self.get_token())
        pass
    # 车型识别
    elif self.comboBox.currentIndex() == 9:
        self.get_car(self.get_token())
        pass
    # Logo 识别
    elif self.comboBox.currentIndex() == 10:
        self.get_logo(self.get_token())
        pass
    pass
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

19.6.4 图像识别

图像识别使用的是百度 AI 接口。访问百度 AI 接口，返回相应的数据，使用 JSON 进行处理。以银行卡识别为例，代码如下：

```
# 银行卡识别
def get_bankcard(self, access_token):
    request_url = "https://aip.baidubce.com/rest/2.0/ocr/v1/bankcard"
    # 以二进制方式打开图片文件
    f = self.get_file_content(self.download_path[0])
    img = base64.b64encode(f)
    params = {"image": img}
    params = urllib.parse.urlencode(params).encode('utf-8')
    request_url = request_url + "?access_token=" + access_token
    request = urllib.request.Request(url=request_url, data=params)
    request.add_header('Content-Type', 'application/x-www-form-urlencoded')
    response = urllib.request.urlopen(request)
    content = response.read()
    if content:
        # 解析返回数据
        bankcards = json.loads(content)
        # 输出返回结果
        print(bankcards)
        strover = '识别结果：\n'
        # 捕捉异常，判断是否正确返回信息
        try:
            if bankcards['result']['bank_card_type']==0:
                bank_card_type='不能识别'
            elif bankcards['result']['bank_card_type']==1:
                bank_card_type = '借记卡'
            elif bankcards['result']['bank_card_type'] == 2:
                bank_card_type = '信用卡'
            strover += '  卡号：{} \n  银行：{} \n  类型：{} \n'\
                .format(bankcards['result']['bank_card_number'],
                        bankcards['result']['bank_name'],bank_card_type)
        # 在错误的时候提示错误原因
        except BaseException:
            error_msg = bankcards['error_msg']
            strover += '  错误：\n {} \n '.format(error_msg)
        # 设置识别显示结果
        self.label_3.setText(strover)
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

19.6.5　复制识别结果到剪贴板

通过上面的步骤，我们已经获取到了图像识别的结果，接下来要实现复制识别结果到剪贴板，该功能在 Python 中很容易实现。

（1）为按钮添加单击事件。代码如下：

```
# 为按钮设置单击方法
self.pushButton_2.clicked.connect(self.copyText)
```

（2）创建 copyText()方法，在该方法中要实现复制识别结果到剪贴板。代码如下：

```
# 复制文字到剪贴板的方法
def copyText(self):
    # 复制文字到剪贴板
    clipboard = QApplication.clipboard()
    # 设置复制的内容
    clipboard.setText(self.label_3.text())
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 20 章

微信跳一跳游戏的辅助工具

20.1 项目概述

最近微信中有一款游戏比较火爆，那就是微信跳一跳游戏，如图 20.1 所示。该游戏需要根据手指按压屏幕的时间控制棋子跳跃到另一个棋盘上，并获取相应的分数，如图 20.2 所示。如果按压的时间过长或者过短，那么棋子就会跳到棋盘的外面，如图 20.3 所示。笔者有些手笨，每次只能跳到 100 分左右，看着微信好友多数都能跳到几百分，心里有些不服气。作为一名开发人员，只能通过代码来助力了，所以本章将通过 Python 实现一款微信跳一跳游戏的辅助工具。

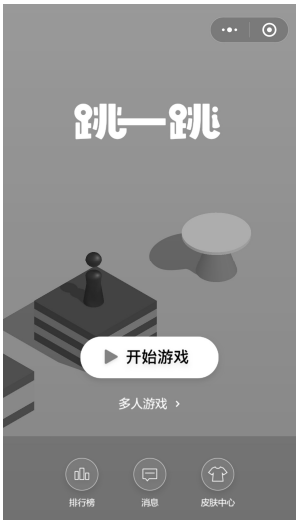


图 20.1 微信跳一跳游戏

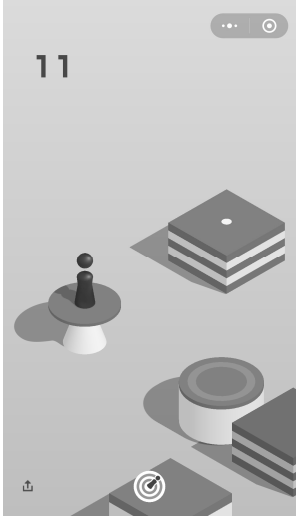


图 20.2 微信跳一跳玩法

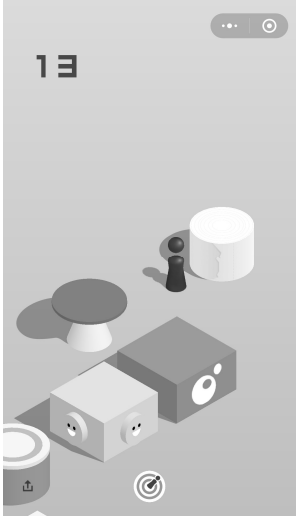


图 20.3 跳跃失败

注意：该辅助工具仅支持 Android 手机中的微信跳一跳小程序。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

20.1.1　设计流程

微信跳一跳游戏辅助工具的设计流程如图 20.4 所示。

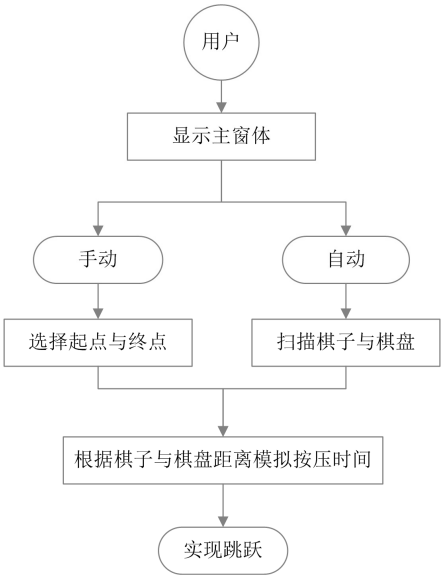


图 20.4　微信跳一跳游戏辅助工具的设计流程

20.1.2　系统预览

微信跳一跳游戏辅助工具的主窗体效果如图 20.5 所示。手动选择起点与终点的效果如图 20.6 所示。开启自动模式后，在控制台中输入需要自动跳跃的次数，如图 20.7 所示。跳跃后自动同步手机画面的效果如图 20.8 所示。

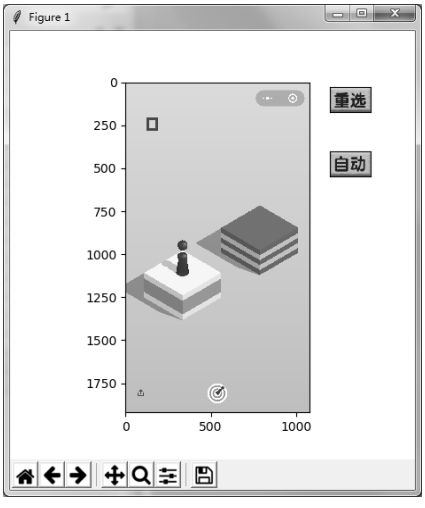


图 20.5　微信跳一跳游戏辅助工具的主窗体效果

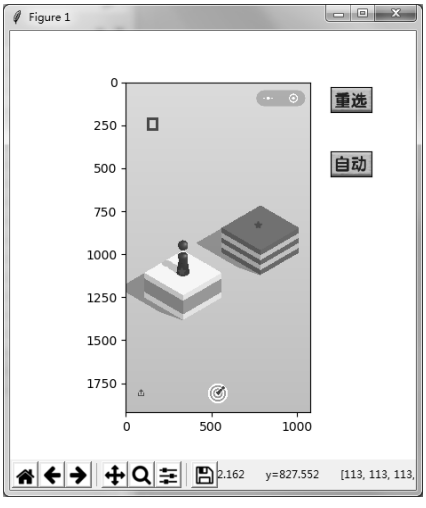


图 20.6　手动选择起点与终点的效果

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

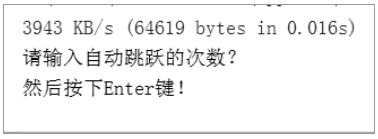


图 20.7　输入自动跳跃的次数

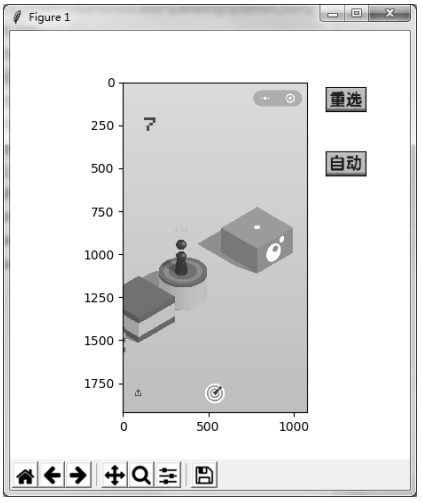


图 20.8　跳跃后自动同步手机画面的效果

20.1.3　开发工具准备

- 操作系统：Windows 7、Windows 8、Windows 10 或 Ubuntu。
- 开发工具：PyCharm。
- 开发模块：PIL、matplotlib。
- 安卓调试工具：ADB 工具。

20.2　开发前的准备工作

在开发微信跳一跳游戏的辅助工具时，主要使用了 PIL 与 matplotlib 模块。其中，PIL 是图像处理模块，包括打开图像文件、切片、旋转、滤镜、输出文字、调色板等工具。该模块仅支持到 Python 2.7 版本。由于开发者对 Python 3.x 版本的使用率越来越高，于是志愿者就在 PIL 模块的基础上开发了兼容版本 Pillow，并且加入了许多新的特性。通过 PIL 模块模糊处理前后的图像效果如图 20.9 所示。

图 20.9　通过 PIL 模块模糊处理前后的图像效果

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

matplotlib 是 Python 的一个 2D 绘图模块。使用该模块，开发者仅需几行代码即可实现绘制直方图、折线图、功率谱、条形图、错误图、散点图等。通过 matplotlib 模块实现的折线图效果如图 20.10 所示。

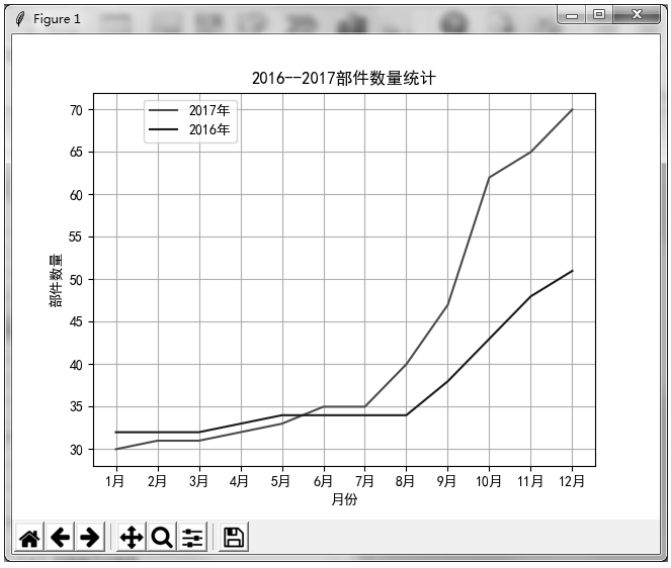


图 20.10　通过 matplotlib 模块实现的折线图效果

20.2.1　PIL 模块的安装

在命令行窗口中输入 pip install pillow 命令，然后按下<Enter>键，将显示如图 20.11 所示的安装进度。

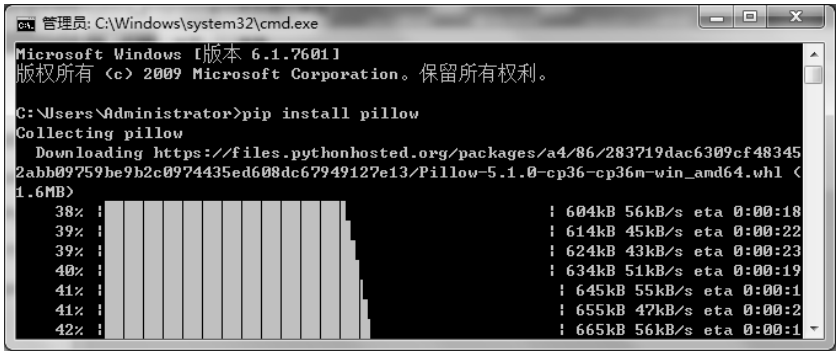


图 20.11　安装 PIL 模块

20.2.2　PIL 模块的使用

PIL 模块中包含各种用于处理图像的工具类，其中比较常用的工具类如表 20.1 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

表 20.1　PIL 模块中常用的工具类

工 具 类	描　　述
Image	用于加载图像及创建新的图像等
ImageChops	用于各种特效效果、图像合成、算法绘画等
ImageColor	该类包含颜色表和从 CSS3 样式颜色说明符到 RGB 元组的转换器
ImageDraw	该类为 Image 对象提供简单的 2D 图形。可以使用此模块创建新图像，对现有图像进行注释或修饰，以及即时生成图形供 Web 使用
ImageFile	该类中包含预定义的一组过滤器，可以与 Image.filter()方法一起使用
ImageFont	用于存储位图字体，并且与 PIL.ImageDraw.Draw.text()方法一起使用
ImageGrab	用于将屏幕或剪贴板上的内容复制到 PIL 图像存储器
ImageMath	用于评估“图像表达式”。该类提供了一个单独的 eval 函数，它接收一个表达式字符串和一幅或多幅图像
ImagePath	用于存储和操作二维矢量数据，路径对象可以传递给 ImageDraw 上的方法
ImageQt	该类包含从 PIL 图像创建 PyQt4、PyQt5 或 PySide QImage 对象的支持
ImageSequence	用于迭代图像序列的帧
ImageStat	用于计算图像或图像区域的全局统计信息
ImageTk	该类包含支持从 PIL 图像创建和修改 Tkinter BitmapImage 和 PhotoImage 对象
ImageFilter	该类为图像过滤器，用于实现滤镜模糊等效果

说明：由于本节的重点是学习开发微信跳一跳游戏的辅助工具的开发思想与实现步骤，所以 PIL 模块的详细使用方法可以参考官方的开发文档进行学习。

20.2.3　matplotlib 模块的使用

matplotlib 模块的安装方法与 PIL 模块的安装方法类似，只是在命令行窗口中输入的命令不同。安装 matplotlib 模块的命令为 pip install matplotlib，然后按下<Enter>键将显示安装进度。

matplotlib 模块中常用的工具类如表 20.2 所示。

表 20.2　matplotlib 模块中常用的工具类

工 具 类	描　　述
matplotlib.pyplot	该类允许快速生成多种绘图方式，如折线图、条形图、散点图等
matplotlib.animation	该类为动画类，其中包含两个子类：一个是 FuncAnimation，该子类是通过反复调用一个函数来制作动画 func 的；另一个是 ArtistAnimation，该子类是使用一组固定的 Artist 对象进行动画制作的
matplotlib.collections	用于高效绘制大量共享大多数属性的对象的类，如大量的线段或多边形
matplotlib.colorbar	具有两个类和一个函数的 Colorbar 工具包。 ColorbarBase：具有完整的彩条绘图功能的基类。它可以按原样为给定的颜色表制作一个颜色条；不需要可映射的对象（如图像） Colorbar：用于图像或等高线图的派生类 make_axes()：用于调整轴大小并添加适合彩条的第二个轴的功能
matplotlib.colors	用于将数字或颜色参数转换为 RGB 或 RGBA

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

工 具 类	描　　述
matplotlib.dates	matplotlib 提供了复杂的日期绘图功能
matplotlib.dviread	用于读取 TeX 输出的 dvi 文件
matplotlib.gridspec	用于指定图中子图的位置
matplotlib.image	支持基本的图像加载、缩放和显示操作
matplotlib.lines	包含各种线条样式、标记和颜色绘制的 2D 线条类
matplotlib.style	用于设置使用的样式
matplotlib.table	用于在 *X* 位置下方放置一张表格

> **说明：**由于 matplotlib 模块的功能比较强大，所以该模块的详细使用方法可以参考官方的开发文档进行学习。

20.2.4　ADB 工具

简单来说，ADB（Android Debug Bridge，安卓调试桥）就是一个命令行窗口，通过在该窗口中输入命令来实现 PC 端与手机模拟器或手机设备进行交互。ADB 工具不需要安装，它是一个压缩工具包，解压后如图 20.12 所示，在该文件夹内只需要将方框内的 adb.exe、AdbWinApi.dll、AdbWinUsbApi.dll 三个文件复制到自己创建的 Python 项目文件夹当中即可。

图 20.12　复制 ADB 文件

复制完成后，在 Python 项目文件夹中的空白处按住<Shift>键的同时单击鼠标右键，在弹出的快捷菜单中选择“在此处打开命令窗口”命令，如图 20.13 所示。

图 20.13　打开命令行窗口

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

将手机连接到计算机，确保自己的计算机中已经安装了手机的驱动程序，并且需要保证计算机与手机可以正常通信。在命令行窗口中输入 adb devices 命令，然后按下<Enter>键，查看手机与计算机是否连接成功，如图 20.14 所示。

图 20.14　查看手机与计算机是否连接成功

20.3　微信跳一跳游戏辅助工具的实现

20.3.1　窗体的实现

在创建微信跳一跳游戏辅助工具的窗体时，需要考虑 3 个方面：第一，预留主窗体的中心位置，用于同步显示手机屏幕中的画面；第二，设置重选按钮，因为在手动选择起点时难免出现选择错误的现象，此时可以通过该按钮重新选择起点的位置；第三，设置自动按钮，用于调用自动跳跃功能。实现的具体步骤如下。

（1）首先创建一个名为 Jump 的文件夹，用于保存微信跳一跳游戏辅助工具的项目文件。然后在该文件夹中创建一个名为 image 的文件夹，用于保存窗体按钮的背景图片。再将 ADB 工具包中的 adb.exe、AdbWinApi.dll、AdbWinUsbApi.dll 三个文件复制到 Jump 文件夹内。最后在 Jump 文件夹内创建 jump.py 文件，在该文件中实现微信跳一跳游戏辅助工具的代码。项目文件结构如图 20.15 所示。

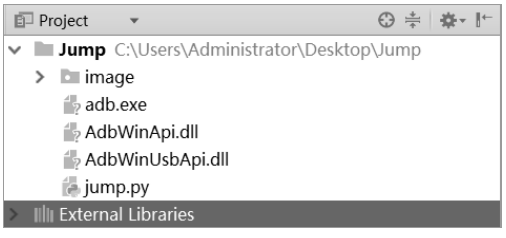


图 20.15　项目文件结构

（2）导入 PIL 图像处理模块与 matplotlib 的画图模块，以及 matplotlib.widgets 中的按钮模块。代码如下：

```
01 import PIL,numpy                              # 图像处理模块
02 import matplotlib.pyplot as plt               # 画图模块
03 from matplotlib.widgets import Button         # 按钮模块
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
04 import warnings                                    # 屏蔽警告
05 warnings.filterwarnings("ignore")
```

（3）判断程序入口，在程序入口代码的下方首先创建一个空白的图形对象；然后设置重选按钮和自动按钮的位置与大小，并且为这两个按钮指定相应的背景图片；最后通过 matplotlib.pyplot 的 show()方法显示主窗体。代码如下：

```
01 if __name__ == "__main__":
02     figure = plt.figure()                                  # 创建一个空白的图形对象
03     # 重选按钮的位置与大小
04     reelect_button_position = plt.axes([0.79, 0.8, 0.1, 0.08])
05     m = numpy.array(PIL.Image.open('image/bt.png')) # 重选按钮的背景图片
06     # 设置重选按钮的背景图片
07     reelect_button = Button(reelect_button_position, label='', image=m)
08     m1 = numpy.array(PIL.Image.open('image/bt1.png'))  # 自动按钮的背景图片
09     # 自动按钮的位置与大小
10     auto_button_position = plt.axes([0.79, 0.65, 0.1, 0.08])
11     # 设置自动按钮的背景图片
12     auto_button = Button(auto_button_position, label='', image=m1)
13     plt.show()  # 显示主窗体
```

微信跳一跳游戏辅助工具的主窗体运行效果如图 20.16 所示。

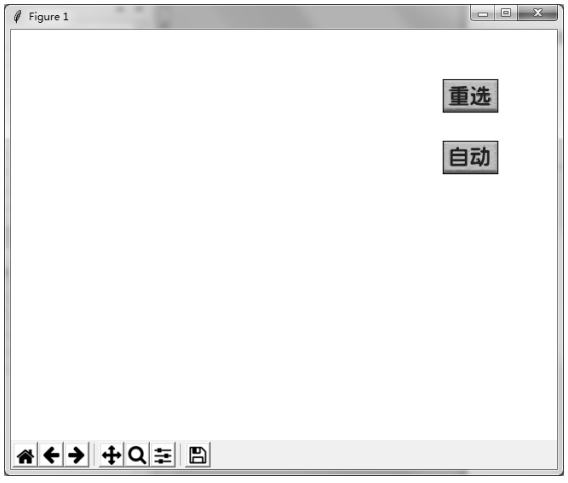


图 20.16　微信跳一跳游戏辅助工具的主窗体运行效果

20.3.2　同步手机屏幕的画面

在实现将手机屏幕的画面同步到辅助工具的窗体时，首先需要使用 ADB 工具中的系统命令获取当前手机屏幕的截图，并将该截图保存在手机的 SD 卡当中。然后继续通过 ADB 工具中的系统命令将截图保存到项目文件夹当中。因为主窗体只能识别转换后的图片，所以需要将打开的图片转换为多维数组并返回。最后将获取的手机屏幕截图显示在主窗体上面即可。实现的具体步骤如下。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（1）在程序入口代码的上面，导入 os 系统模块，并创建 get_screen_image()方法。在该方法中，首先通过 ADB 工具中的系统命令获取手机屏幕的截图并将截图保存到手机的 SD 卡当中，然后将手机中的截图保存到当前项目的文件夹当中，最后将打开的图片转换为多维数组并返回。代码如下：

```
import os  # 系统模块

# 获取手机屏幕截图
def get_screen_image():
    # 通过 ADB 工具中的系统命令截取当前手机屏幕，然后将截图保存在手机的 SD 卡根目录下
    os.system('adb shell screencap -p /sdcard/screen.png')
    # 将手机中的截图保存在当前项目的文件夹当中
    os.system('adb pull /sdcard/screen.png')
    # 将打开的图片转换为多维数组并返回
    return numpy.array(PIL.Image.open('screen.png'))
```

（2）在程序入口代码中创建空白图形对象代码的下面，将手机屏幕的截图显示在主窗体当中。代码如下：

```
# 将获取的图片显示在主窗体当中
axes_image = plt.imshow(get_screen_image(), animated=True)
```

同步手机屏幕画面的运行效果如图 20.17 所示。

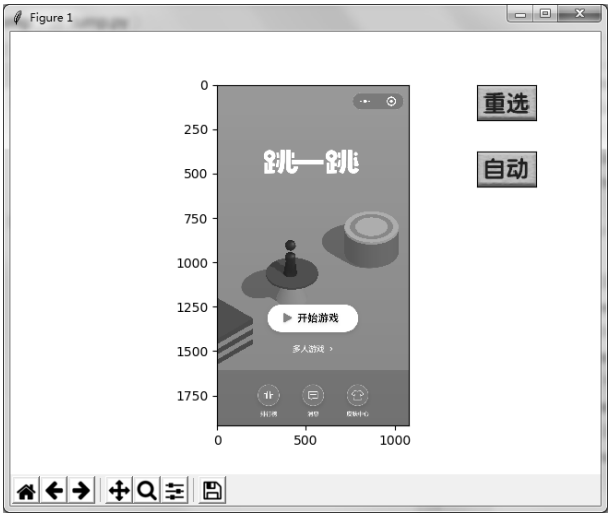


图 20.17　同步手机屏幕画面的运行效果

> **注意：**在实现了同步手机屏幕画面后，在项目文件结构中会自动添加一幅手机屏幕截图，如图 20.18 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

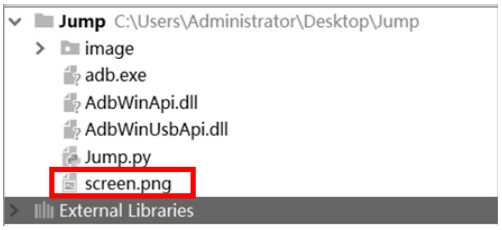


图 20.18　自动添加的手机屏幕截图

20.3.3　选择起点与终点实现跳跃

在手动选择起点与终点实现跳跃时，首先需要设置主窗体的单击事件；然后获取鼠标在截图中选中起点与终点的坐标，通过两个坐标点的距离计算手指需要在手机屏幕上按压的时间；最后通过 ADB 工具中的系统命令与计算出的按压时间，模拟在手机屏幕上进行按压操作，从而实现棋子的跳跃动作。在棋子跳跃完成后，需要在主窗体中进行手机画面的更新。实现的具体步骤如下。

（1）在 get_screen_image()方法的上面，首先导入时间模块与子线程模块，然后定义一个用于保存选中位置坐标的列表，再定义一个子视图变量用于在选中的位置画点，最后定义一个是否启动自动模式的标记。代码如下：

```
01 import time          # 时间模块
02 import threading     # 子线程模块
03
04 coor = []            # 保存选中位置的坐标
05 ax = None            # 子视图变量
06 isAuto = False       # 标记是否启动了自动模式
```

（2）在程序入口代码中将获取的图片显示在主窗体的代码下面，设置主窗体的单击事件并设置单击事件所执行的方法名称为 on_click。代码如下：

```
figure.canvas.mpl_connect('button_press_event', on_click)  # 设置主窗体的单击事件
```

> **说明：**在编写步骤（2）中的代码时，on_click 方法名称的下面将出现红色波浪线，这是因为还没有创建名为 on_click 的方法。此处不需要理会，在接下来的步骤中会创建该方法，红色波浪线将消失。

（3）在 get_screen_image()方法的下面创建 on_click()方法，在该方法中首先获取选中的起点与终点坐标，然后将这两个坐标点添加到 coor 列表中。同时在选择位置的坐标点上绘制红色的*号，当 coor 列表中已经保存了起点与终点的坐标时调用计算距离的方法并实现跳跃，跳跃完成后调用更新画面的方法实现主窗体手机画面的更新。代码如下：

```
01 # 坐标轴单击事件，event 用于获取单击坐标点，coor 用于保存单击的起始坐标点与结束坐标点
02 def on_click(event):
03     if isAuto == False:   # 判断是否启动自动模式
04         if event.xdata != None and event.ydata != None: # 判断获取的坐标点是否为空
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            # 获取单击的坐标点并转换为 float 类型
            x = float(event.xdata)
            y = float(event.ydata)
            # 此处判断为了躲避重选与自动按钮的坐标
            if x > 70 and y > 70:
                # 将单击坐标轴上的 x、y 坐标点添加到 coor 列表中
                coor.append((x, y))
                if len(coor) == 1:
                    print('选中起点')
                else:
                    print('选中终点')
                global ax
                ax = figure.add_subplot(1, 1, 1)  # 添加一个子图，也就是绘制我们选中的点
                ax.plot(x, y, 'r*')   # 绘制红色的*号
                figure.canvas.draw()  # 重画
            # 如果 coor 列表的长度等于 2，则说明已经获取了单击的起始位置与结束位置的坐标点
                if len(coor) == 2:
                    # 调用计算方法，将起始位置与结束位置的坐标点传递过去，然后清空坐标点
                    jump_to_next(coor.pop(), coor.pop())
                    ax.lines.clear()  # 清除选中的绘制点
                    # 通过线程进行更新
                    th = threading.Thread(target=update)
                    th.start()
    else:
        print('已开启自动模式！')
```

（4）在 on_click()方法的下面创建 jump_to_next()方法，该方法用于计算手动选择的起点与终点的距离，然后通过 ADB 工具中的系统命令与计算出的按压时间，模拟在手机屏幕上进行按压实现棋子的跳跃功能。代码如下：

```
# 实现跳跃的方法
def jump_to_next(point1, point2):  # 计算弦的长度
    x1, y1 = point1;  # 起点坐标
    x2, y2 = point2;  # 终点坐标
    # 计算起点与终点的距离
    distance = ((x2 - x1) ** 2 + (y2 - y1) ** 2) ** 0.5
    # 执行按压手机屏幕的命令，每个像素点 1.35 毫秒
    os.system('adb shell input swipe 550 1550 550 1550 {}'.format(int(distance *
1.35)))
    print('跳')
```

（5）在 jump_to_next()方法的下面创建 update()方法，该方法用于更新主窗体手机画面。代码如下：

```
# 更新主窗体手机画面
def update():
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    time.sleep(0.8)  # 休眠 0.8 秒
    print('更新')
    # 更新图片
    axes_image.set_array(get_screen_image())
    figure.canvas.draw()  # 重画
```

运行程序后，选中起点与终点，如图 20.19 所示，棋子将自动跳跃到棋盘的指定位置，如图 20.20 所示。

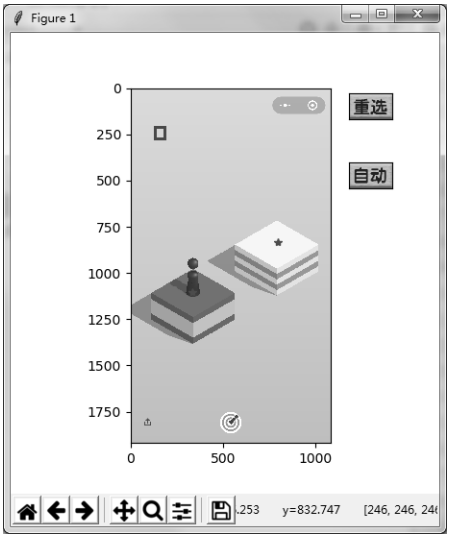


图 20.19　选中起点与终点

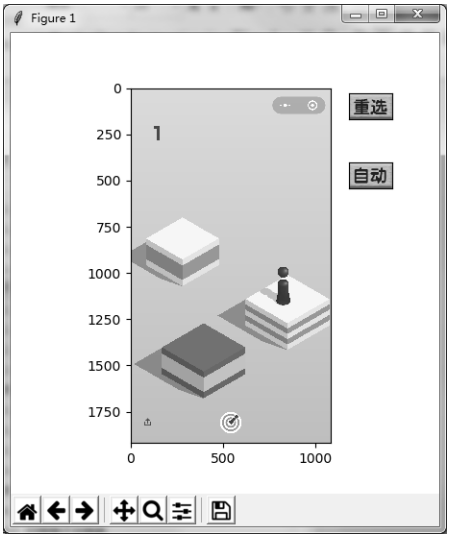


图 20.20　棋子自动跳跃到棋盘的指定位置

20.3.4　实现重选功能

为了避免手动选择起点位置时出现错误，在本节中添加了重选功能。实现的具体步骤如下。

（1）在程序入口代码中设置重选按钮背景图片代码的下面，设置重选按钮的单击事件并设置单击事件所执行的方法名称为 button_click。代码如下：

```
reelect_button.on_clicked(button_click)  # 设置重选按钮的单击事件
```

（2）在 update()方法的下面创建 button_click()方法，在该方法中首先清空保存选中坐标点的 coor 列表，然后清除选中的绘制点，最后进行重画。代码如下：

```
# 重选按钮的单击事件处理
def button_click(event):
    if len(coor) < 2 and ax != None:
        coor.clear()              # 清空选中的坐标点
        ax.lines.clear()          # 清除选中的绘制点
        figure.canvas.draw()      # 重画
```

运行程序后，选择起点如图 20.21 所示，然后单击“重选”按钮将清除选择的起点*号，如图 20.22 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

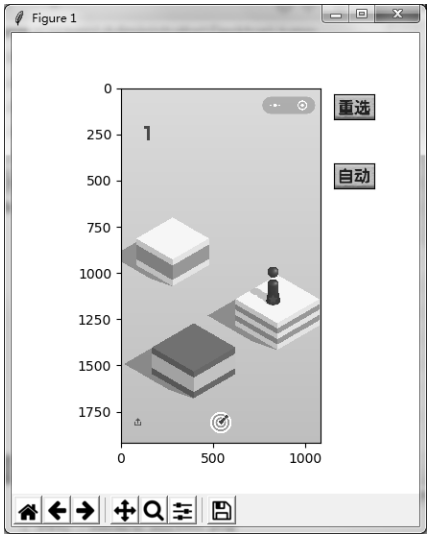


图 20.21　选择起点

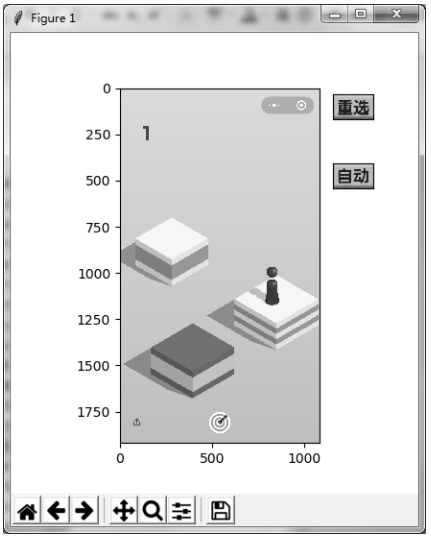


图 20.22　重选起点

20.4　微信跳一跳游戏辅助工具（自动）的实现

20.4.1　同步手机屏幕的画面

在实现微信跳一跳辅助工具的自动模式时，同样需要同步手机屏幕的画面，但是此处将通过另一种方法来实现这一功能。在实际开发中实现一种功能时，不能固定地使用一种模块或一种方式来进行开发，需要尽量想出不同的解决方案，从中选取最优秀的方案进行功能的实现。实现的具体步骤如下。

（1）在 Jump 项目文件夹中创建 auto.py 文件，在该文件中首先导入 os 系统模块，然后导入子进程模块，再导入随机数模块，最后定义一个是否循环游戏的 loop 变量。代码如下：

```
01 import os                # 系统模块
02 import subprocess        # 子进程模块
03 import random            # 随机数模块
04
05
06 # 是否循环游戏
07 loop = False
```

（2）在是否循环游戏的变量下面创建 get_screenshot()方法，在该方法中首先通过 subprocess.Popen()方法与 ADB 工具中的系统命令实现手机的截图功能，然后读取截图信息，再将其转换为计算机识别的格式，最后将图片以写入的方式保存在 Jump 项目文件夹当中。代码如下：

```
01 # 获取屏幕截图
02 def get_screenshot():
03     # 截图
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    process = subprocess.Popen('adb shell screencap -p',
                                          shell=True, stdout=subprocess.PIPE)
    if process != None:
        # 读取截图信息
        screenshot = process.stdout.read()
        # 计算机识别格式
        binary_screenshot = screenshot.replace(b'\r\r\n', b'\n')
        # 将图片写入项目文件夹下
        with open('autojump.png', 'wb') as f:
            f.write(binary_screenshot)
            print('截图并将该图片放在项目文件夹内! ')
```

20.4.2　扫描棋子与棋盘

因为是自动模式，所以需要系统通过扫描的方式识别出棋子与棋盘的中心点位置。扫描的方式是通过像素矩阵加载需要扫描的图片，然后系统将通过图片的每个像素点进行扫描。扫描的过程总体分为 3 步。

（1）从上到下以 50px（像素）为单位快速扫描图片，找到第一个棋盘位置，确定该位置为扫描 Y 轴的最大值。

（2）从已经获取的 Y 轴最大值开始详细地扫描，根据棋子特殊的颜色找到棋子最后一行的像素点取中，设置该中心点为棋子底部的中心点。

（3）通过棋子底部的中心点确定寻找棋盘的范围。因为游戏的背景色始终相同，所以可以通过色差的方式找到棋盘并获取棋盘横向坐标的中心点。

实现的具体步骤如下。

（1）在 get_screenshot()方法的下面创建 find_piece_and_board()方法，在该方法中通过快速扫描棋盘的方式获取 Y 轴最大值。代码如下：

```
def find_piece_and_board(img):
    # 获取图片的宽、高
    w, h = img.size
    # 棋子的底边界
    piece_y_max = 0
    scan_x_side = int(w / 8)    # 扫描棋子的左右边界减少开销
    scan_start_y = 0            # 扫描起始 Y 坐标
    # 图片像素矩阵
    img_pixel = img.load()
    if not loop:  # 是否循环游戏
        if sum(img_pixel[5, 5][:-1]) < 150:  # 屏幕变黑色，也就是手机锁屏后游戏结束
            stop()
    # 以 50px 为步长，尝试探测 scan_start_y
    for i in range(int(h / 3), int(h * 2 / 3), 50):
        # 每行第一个像素
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        first_pixel = img_pixel[0, i]
        for j in range(1, w):
            # 每个像素
            pixel = img_pixel[j, i]
            # 如果不是纯色，则说明碰到了新的棋盘，跳出，找到了 Y 轴最大值
            if pixel[0] != first_pixel[0] or pixel[1] \
                            != first_pixel[1] or pixel[2] != first_ pixel[2]:
                # 找到最高点
                scan_start_y = i - 50
                break
        if scan_start_y:
            break
```

（2）在 find_piece_and_board()方法中依次添加代码，用于扫描棋子并找到棋子底部的中心点坐标。代码如下：

```
    # 从上往下开始扫描棋子，棋子位于屏幕上半部分
    left = 0
    right = 0
    for i in range(scan_start_y, int(h * 2 / 3)):
        flag = True
        # 切掉左右 1/8
        for j in range(scan_x_side, w - scan_x_side):
            pixel = img_pixel[j, i]
            # 根据棋子的最低行的颜色判断，找到最后一行像素点的起始和末尾
            # 判断 RGB 颜色为棋子的颜色
          if (50 < pixel[0] < 60) and (53 < pixel[1] < 63) and (95 < pixel[2] < 110):
                if flag:
                    left = j       # 记录棋子底部第一个横坐标
                    flag = False
                right = j          # 记录棋子底部最后的横坐标
                # 棋子最后一行像素的 Y 坐标点
                piece_y_max = max(i, piece_y_max)
    if not all((left, right)):
        return 0, 0, 0, 0
    # 棋子底部中心点
    piece_x = (left + right) // 2
```

> **注意：**由于 find_piece_and_board()方法中的代码较长，所以读者在编写代码时请注意代码的格式，建议参照源码进行代码的编写。

（3）在 find_piece_and_board()方法中依次添加代码，用于扫描棋盘并找到棋盘横向的中心点坐标。代码如下：

```
    # 寻找落点
    board_x = 0
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    # 限制棋盘扫描的横坐标，当棋子位置小于图片宽度的一半时
    # 设置扫描棋盘的范围
    if piece_x < w / 2:
        board_x_start, board_x_end = w // 2, w  # 起点和终点的中点是画面中心
    else:
        board_x_start, board_x_end = 0, w // 2

    # 寻找落点顶点
    board_x_set = []  # 目标坐标集合/改为 list，避免去重
    for by in range((h - w) // 2, (h + w) // 2, 4):
        bg_pixel = img_pixel[0, by]
        for bx in range(board_x_start, board_x_end):
            pixel = img_pixel[bx, by]
            # 色差比较
            if (abs(pixel[0] - bg_pixel[0]) +
                    abs(pixel[1] - bg_pixel[1]) +
                    abs(pixel[2] - bg_pixel[2]) > 10):
                board_x_set.append(bx)

        if len(board_x_set) > 10:
            # 根据筛选出来的色差坐标，获取棋盘横向坐标的中心点
            board_x = sum(board_x_set) / len(board_x_set)
            break  # 找到了退出
    print('读取图片并获取棋子与棋盘位置横向中心坐标')
    return piece_x, board_x
```

20.4.3　实现自动跳跃

找到了棋子与棋盘的中心坐标点就相当于找到了起点与终点。与手动模式不同的是，首先通过获取起点与终点的水平距离和手机屏幕的宽度计算出棋子跳跃的百分比，然后通过棋子跳跃的百分比计算按压的时长，接着通过 ADB 工具中的系统命令实现棋子的跳跃功能，最后通过主窗体中的自动按钮开启自动模式即可。实现的具体步骤如下。

（1）在 find_piece_and_board()方法的下面，首先创建 jump()方法，该方法用于实现棋子的跳跃功能；然后创建 stop()方法，该方法用于在执行棋子自动跳跃功能的过程中，如果将手机锁屏，那么将执行 stop()方法停止自动跳跃。代码如下：

```
def jump(piece_x, board_x, im, point):
    distanceX = abs(board_x - piece_x)      # 起点到目标的水平距离
    shortEdge = min(im.size)                # 屏幕宽度
    jumpPercent = distanceX / shortEdge     # 跳跃百分比
    jumpFullWidth = 1700                    # 跳过整个宽度需要按压的毫秒数
    press_time = round(jumpFullWidth * jumpPercent)  # 按压时长
    press_time = 0 if not press_time else max(
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        press_time, 200)                    # 当 press_time 大于 0 时限定最小值
    # 按压命令
    cmd = 'adb shell input swipe {x1} {y1} {x2} {y2} {duration}'.format(
        x1=point[0],
        y1=point[1],
        x2=point[0] + random.randint(0, 3),
        y2=point[1] + random.randint(0, 3),
        duration=press_time
    )
    os.system(cmd)
    print('生成按压命令，执行跳跃！')
    # 停止自动跳跃
def stop():
    exit('停止自动跳跃！')
```

（2）打开 jump.py 文件，在该文件的程序入口代码中设置自动按钮背景图片代码的下面设置自动按钮的单击事件，并设置单击事件所执行的方法名称为 auto_click。代码如下：

```
auto_button.on_clicked(auto_click)  # 设置自动按钮的单击事件
```

（3）在 button_click()方法的下面创建 auto_click()方法，在该方法中通过线程启动自动模式。代码如下：

```
def auto_click(event):
    # 通过线程启动自动模式
    th = threading.Thread(target=auto_mode)
    th.start()
```

（4）在 auto_click()方法的下面，首先导入 auto 自动跳跃文件与随机数模块，然后创建 auto_mode()方法。在该方法中首先提示用户输入棋子自动跳跃的次数，然后根据棋子与棋盘中心点的坐标和随机生成的按压点实现棋子的跳跃，跳跃后进行主窗体画面的更新，最后设置停留 2 秒后执行下一次跳跃，如果达到输入的跳跃次数则停止跳跃。代码如下：

```
import auto         # 自动跳跃文件
import random       # 随机数模块


# 自动模式
def auto_mode():
    global isAuto
    isAuto = True  # 标记开启自动模式
    print('请输入自动跳跃的次数？')
    print('然后按下 Enter 键！')
    info = input()
    for i in range(int(info)):
        # 获取截图
        auto.get_screenshot()
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        # 生成图片对象
        img = PIL.Image.open('autojump.png')
        # 获取棋子与棋盘位置横向中心坐标
        piece_x, board_x = auto.find_piece_and_board(img)
        # 生成一个随机按压点
        press_point = (random.randint(*[815, 923]),
                       random.randint(*[1509, 1658]))
        # 进行跳跃
        auto.jump(piece_x, board_x, img, press_point)
        update()              # 更新画面
        time.sleep(2)         # 设置停留时间
        if (i + 1) == int(info):
            isAuto = False    # 标记关闭自动模式
            print('结束此次自动模式! ')
```

运行程序后，单击自动按钮，控制台中将提示输入自动跳跃的次数，如图 20.23 所示。然后棋子将根据输入的次数进行自动跳跃，跳跃完成后，控制台中将显示如图 20.24 所示的提示信息。

```
2858 KB/s (64397 bytes in 0.022s)
请输入自动跳跃的次数?
然后按下Enter键!
```

图 20.23　提示输入自动跳跃的次数

```
截图并将该图片放在项目文件夹内!
读取图片并获取棋子与棋盘位置横向中心坐标
生成按压命令，执行跳跃!
更新
4688 KB/s (81621 bytes in 0.017s)
结束此次自动模式!
```

图 20.24　显示跳跃完成后的提示信息

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 21 章

汽车之家图片抓取工具

21.1 项目概述

有时候，我们浏览网站，看到网站上有很多自己喜欢的图片，想保存下来，但一张一张地保存又很麻烦。本章将通过 Python 与 PyQt5 实现一款下载汽车图片的工具。本项目主要爬取网站上阿斯顿·马丁的汽车图片进行下载并显示。

21.2 设计流程

汽车之家图片抓取工具的设计流程如图 21.1 所示。

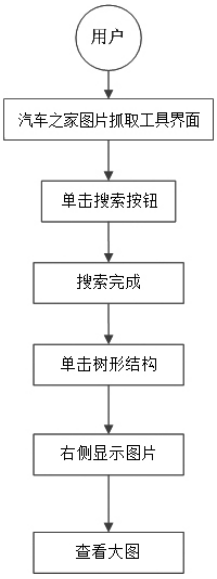


图 21.1　汽车之家图片抓取工具的设计流程

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

21.2.1　系统预览

汽车之家图片抓取工具默认显示效果如图 21.2 所示。

图 21.2　汽车之家图片抓取工具默认显示效果

单击“阿斯顿 • 马丁 汽车图片”按钮，工具开始爬取图片并下载。搜索完成后，左侧树形结构控件出现三角箭头，如图 21.3 所示。

图 21.3　类别选择显示

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

单击树形结构，右侧显示图片列表，如图 21.4 所示。

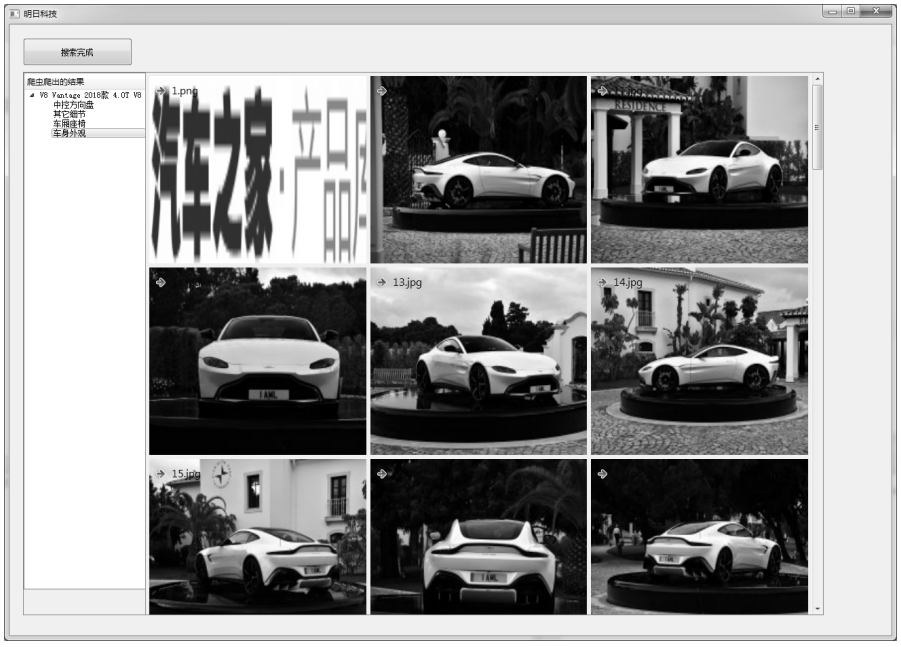


图 21.4　显示图片列表

想要查看哪张图片的大图，可以单击图片左上角的按键，会打开浏览图片界面，如图 21.5 所示。

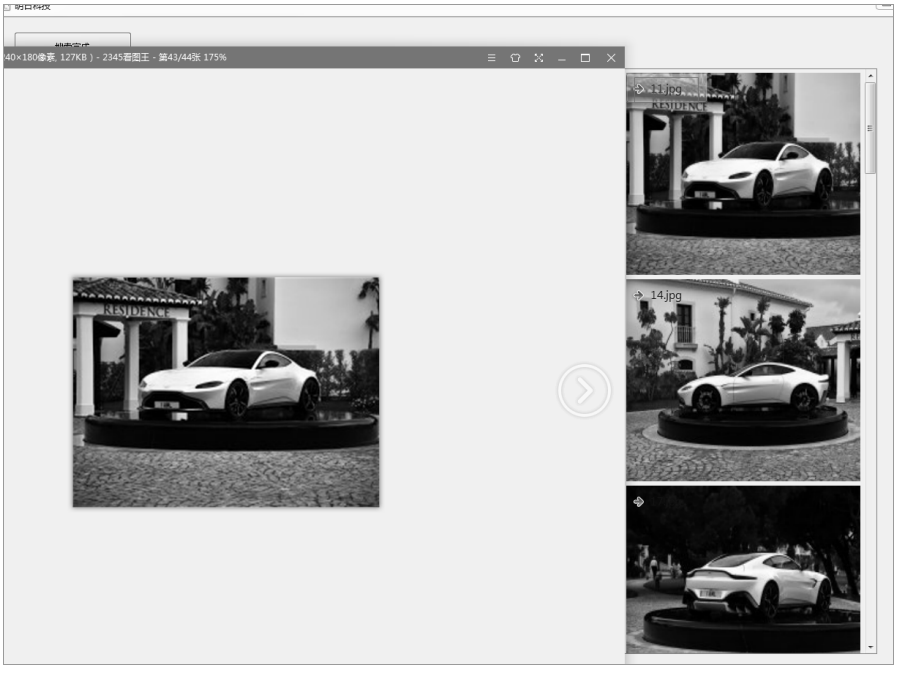


图 21.5　浏览图片界面

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

21.2.2　开发工具准备

- 操作系统：Windows 7、Windows 8、Windows 10 或 Ubuntu。
- 开发工具：PyCharm、Qt。
- 开发模块：urllib、urllib.request、bs4、Image、PyQt5。

21.3　汽车之家图片抓取工具的开发

21.3.1　主窗体设计

根据窗体进行设计，设计完成后保存文件并生成 Python 文件，文件中会自动生成 Python 代码，我们在其中对 UI 进行一些基础设置，包括绑定单击事件、修改按钮显示内容等。代码如下：

```
# UI设置方法 设置ui属性
def retranslateUi(self, Form):
    _translate = QtCore.QCoreApplication.translate
    # 设置窗体名称
    Form.setWindowTitle(_translate("Form", "明日科技"))
    # 设置按钮显示文字
    self.pushButton.setText(_translate("Form", "阿斯顿·马丁 汽车图片"))
    # 设置按钮显示文字
    self.pushButton1.setText(_translate("Form", "搜索完成"))
    # 为按钮添加单击事件
    self.pushButton.clicked.connect(self.btnstate)
    # 获取树形结构根节点
    self.root = QTreeWidgetItem(self.treeView)
    # 在根节点中添加数据
    self.root.setText(0, 'V8 Vantage 2018款 4.0T V8')
```

21.3.2　创建爬虫类

新建爬虫类爬取网站的图片并下载，然后保存到指定的文件夹下。代码如下：

```
class ReTbmm():
    def Retbmm(self):
        # 爬虫开始爬取时间
        start = time.time()
        # 用于返回当前工作目录
        self.cdir = os.getcwd()
        # 爬取的网址：https://www.autohome.com.cn/spec/32890/?pvareaid=2023562
        # 车身外观
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
09        url1 = 'https://car.autohome.com.cn/pic/series-s32890/385-1.html#pvareaid=
2023594'
10        # 中控方向盘
11        url2  =  'https://car.autohome.com.cn/pic/series-s32890/385-10.html#
pvareaid=2023594'
12        # 车厢座椅
13        url3  =  'https://car.autohome.com.cn/pic/series-s32890/385-3.html#
pvareaid=2023594'
14        # 其他细节
15        url4  =  'https://car.autohome.com.cn/pic/series-s32890/385-12.html#
pvareaid=2023594'
16        self.getImg('车身外观', url1)
17        self.getImg('中控方向盘', url2)
18        self.getImg('车厢座椅', url3)
19        self.getImg('其他细节', url4)
20        end = time.time()
21        # 输出运行时间
22        print("run time:" + str(end - start))
23    # 下载图片方法
24    def getImg(self, name, urls):
25        user_agent = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
537.36 (KHTML, like Gecko) Chrome/42.0.2311.135 Safari/537.36 Edge/12.10240'
26        headers = {'User-Agent': user_agent}
27        # 访问链接
28        request = urllib.request.Request(urls, headers=headers)
29        # 获取数据
30        response = urllib.request.urlopen(request)
31        # 解析数据
32        bsObj = BeautifulSoup(response, 'html.parser')
33        # 查找所有 img 标记
34        t1 = bsObj.find_all('img')
35        for t2 in t1:
36            t3 = t2.get('src')
37            print(t3)
38        # 创建图片路径
39        path = self.cdir + '/mrsoft/' + str(name)
40        # 读取路径
41        if not os.path.exists(path):
42            # 根据路径建立图片文件夹
43            os.makedirs(path)
44        # 每次调用初始化图片序号
45        n = 0
46        # 循环图片集合
47        for img in t1:
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            # 每次图片顺序加 1
            n = n + 1
            # 获取图片路径
            link = img.get('src')
            # 判断图片路径是否存在
            if link:
                # 拼接图片链接
                s = "http:" + str(link)
                # 分离文件扩展名
                i = link[link.rfind('.'):]
                try:
                    # 访问图片链接
                    request = urllib.request.Request(s)
                    # 获取返回事件
                    response = urllib.request.urlopen(request)
                    # 读取返回内容
                    imgData = response.read()
                    # 创建文件
                    pathfile = path + r'/' + str(n) + i
                    # 打开文件
                    with open(pathfile, 'wb') as f:
                        # 将图片写入文件
                        f.write(imgData)
                        # 图片写入完成关闭文件
                        f.close()
                        print("thread " + name + " write:" + pathfile)
                except:
                    print(str(name) + " thread write false:" + s)
```

21.3.3　开启爬虫类

在搜索按钮的事件中开启爬虫类，进行汽车图片的下载，并对树形控件进行处理，添加文件夹名称到树形结构中。

（1）开始爬虫事件，并添加文件夹名称到树形结构中充当二级栏目。代码如下：

```
# 搜索方法
def btnstate(self):
    # 开始搜索 隐藏按钮
    self.pushButton.setVisible(False)
    # 实例化爬虫类
    ui = ReTbmm()
    # 开启爬虫方法
    ui.Retbmm()
    # 显示已完成按钮
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    self.pushButton1.setVisible(True)
    # 设置文件夹路径，为树形结构做准备
    self.path = cdir + '/mrsoft'
    # 查找路径下的所有文件夹名称
    dirs = os.listdir(self.path)
    # 循环文件夹名称
    for dir in dirs:
        # 添加文件夹名称到树形结构中
        QTreeWidgetItem(self.root).setText(0, dir)
    self.treeView.clicked.connect(self.onTreeClicked)
```

（2）处理树形结构单击事件后，单击每个子栏目，在右侧显示子栏目文件的图片。代码如下：

```
#  树形结构单击后在这里处理
def onTreeClicked(self, Qmodelidx):
    # 获取单击的树形结构
    items = self.treeView.currentItem()
    # 判断单击的节点
    if items.text(0) == 'V8 Vantage 2018款 4.0T V8':
        # 删除节点root下的子节点
        self.root.takeChildren()
        # 获取路径下的所有文件
        dirs = os.listdir(self.path)
        # 循环文件
        for dir in dirs:
            # 设置子节点
            QTreeWidgetItem(self.root).setText(0, dir)
        # 注册单击事件
        self.treeView.clicked.connect(self.onTreeClicked)
        pass
    else:
        # 每次单击循环删除管理器的组件
        while self.gridLayout.count():
            # 获取第一个组件
            item = self.gridLayout.takeAt(0)
            # 删除组件
            widget = item.widget()
            widget.deleteLater()
        # 每次单击树形结构把图片集合清空
        filenames = []
        # 根据路径查找文件夹下的所有文件，循环文件夹下的文件名称
        # listdir的参数是文件夹的路径
        for filename in os.listdir(cdir + '/mrsoft/' + items.text(0)):
            # 把名称添加到集合中
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
32              filenames.append(filename)
33          # 行数标记
34          i = -1
35          # 根据图片的数量进行循环
36          for n in range(len(filenames)):
37              # x 确定每行显示的个数，每行 3 个
38              x = n % 3
39              # 当 x 为 0 的时候设置换行，行数+1
40              if x == 0:
41                  i += 1
42              # 创建布局
43              self.widget = QWidget()
44              # 设置布局大小
45              self.widget.setGeometry(QtCore.QRect(110, 40, 350, 300))
46              # 给布局命名
47              self.widget.setObjectName("widget" + str(n))
48              # 创建一个 Qlabel 控件用于显示图片，设置控件在 QWidget 中
49              self.label = QLabel(self.widget)
50              # 设置大小
51              self.label.setGeometry(QtCore.QRect(0, 0, 350, 300))
52              # 设置要显示的图片
53              self.label.setPixmap(QPixmap(self.path + '/' + items.text(0) + '/' +
filenames[n]))
54              # 图片显示方式，让图片适应 QLabel 的大小
55              self.label.setScaledContents(True)
56              # 给图片控件命名
57              self.label.setObjectName("label" + str(n))
58              # 创建按钮用于单击后放大图片，设置按钮在 QWidget 中
59              self.commandLinkButton = QCommandLinkButton(self.widget)
60              # 设置按钮位置
61              self.commandLinkButton.setGeometry(QtCore.QRect(0, 0, 111, 41))
62              # 给按钮命名
63              self.commandLinkButton.setObjectName("label" + str(n))
64              # 设置按钮上的显示文字
65              self.commandLinkButton.setText(filenames[n])
66              # 注册信号槽，使用 lambda 传递参数给方法
67              self.commandLinkButton.clicked.connect(lambda: self.wichbtn(self.
path + '/' + items.text(0) + '/'))
68             # 把动态添加的 widegt 布局添加到 gridLayout 中，i、x 分别代表行数及每行显示的个数
69              self.gridLayout.addWidget(self.widget, i, x)
70          # 设置上下滑动控件可以滑动，把 scrollAreaWidgetContents_2 添加到 scrollArea 中
71          self.scrollArea_2.setWidget(self.scrollAreaWidgetContents_2)
72          self.verticalLayout.addWidget(self.scrollArea_2)
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
73          # 设置 scrollAreaWidgetContents_2 最大宽度为 scrollArea_2 宽度，都可以显示出
来，不用左右滑动
74          self.scrollAreaWidgetContents_2.setMinimumWidth(800)
75          # 设置高度为动态高度，根据行数确定高度，每行 500 像素
76          self.scrollAreaWidgetContents_2.setMinimumHeight(i * 300)
```

21.3.4 查看大图功能

查看大图功能通过 PIL 模块中的 Image 实现，代码如下：

```
01 # 信号槽，单击按钮显示大图功能
02 def wichbtn(self, tppath):
03     # 获取信号源需要单击的按钮
04     sender = self.gridLayout.sender()
05     # 使用计算机中的看图工具打开图片
06     img = Image.open(tppath + sender.text())
07     img.show()
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

第 22 章

巡店宝（企业商品运营利器）

22.1 项目概述

如果想在京东商城中查询某本图书的相关信息，则通常是通过在京东商城的官方网站或者 App 应用中搜索需要查询的图书名称，然后才可以查看图书的相关信息的。例如，在京东商城的官方网站中查询图书信息，如图 22.1 所示。

图 22.1　在京东商城的官方网站中查询图书信息

以上查询方式有些笨拙，本节将通过 Python 爬虫实现一个爬取京东图书信息的查询工具——巡店宝（京东版）。

注意：爬虫类的辅助工具仅供学习参考，不得用于商业用途。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

22.2 设计流程

巡店宝（京东版）查询工具的设计流程如图 22.2 所示。

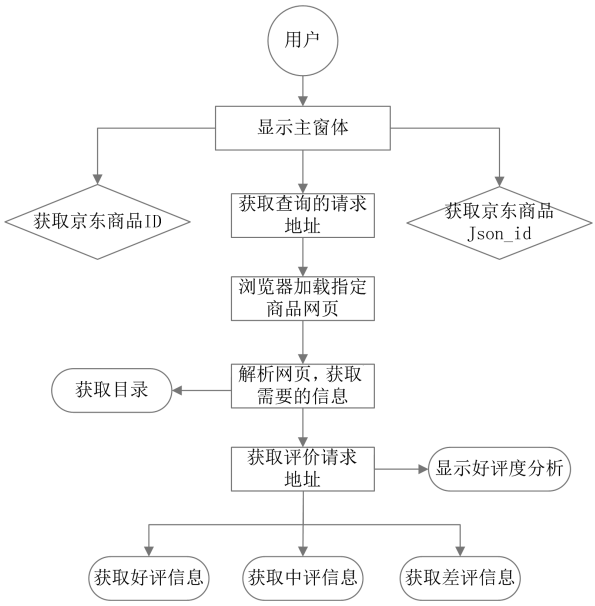


图 22.2 巡店宝（京东版）查询工具的设计流程

22.3 系统预览

巡店宝（京东版）查询工具的主窗体效果如图 22.3 所示。

图 22.3 巡店宝（京东版）查询工具的主窗体效果

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

在编辑框中输入指定的图书名称，然后单击“查询”按钮，（根据当前网络速度）等待一定的时间，即可显示如图 22.4 所示的图书信息。默认显示好评信息，选择其他单选按钮即可显示中评与差评信息。

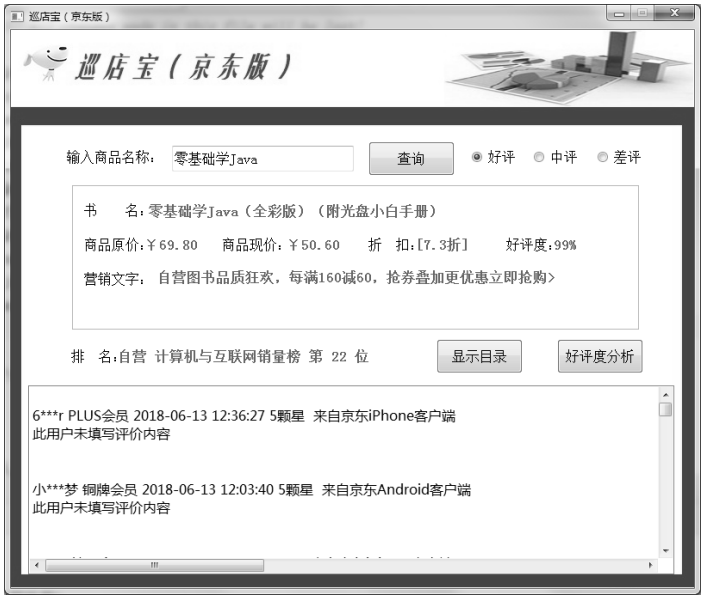


图 22.4　显示图书信息

图书信息显示完成后，单击“显示目录”按钮，将显示如图 22.5 所示的图书目录。单击“好评度分析”按钮，即可显示如图 22.6 所示的好评对比图。

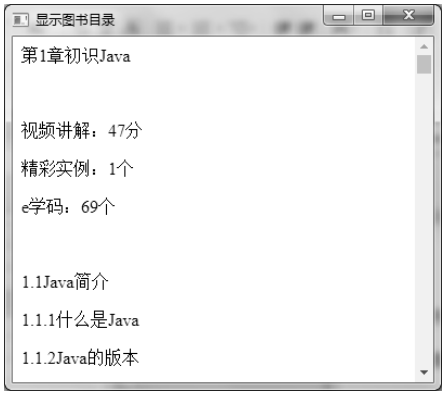


图 22.5　显示图书目录

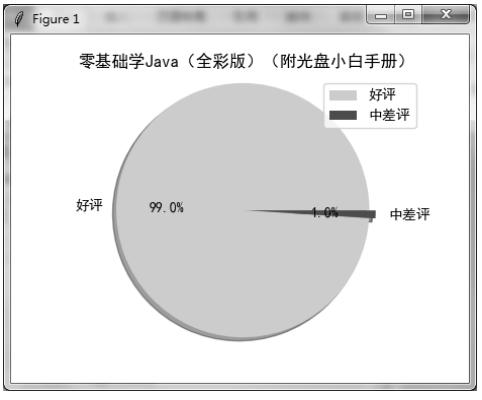


图 22.6　显示好评对比图

22.4　开发工具准备

- 操作系统：Windows 7、Windows 8、Windows 10 或 Ubuntu。
- 开发工具：PyCharm。
- 开发模块：PyQt5、matplotlib、selenium、BeautifulSoup、requests。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

- 谷歌浏览器：66.0.3359.139（正式版本）（64 位）。
- 谷歌浏览器驱动：chromedriver.exe（2.38）。

22.5 开发前的准备工作

在开发京东图书信息查询工具时，主要使用了 PyQt5、matplotlib、selenium、BeautifulSoup、requests 等模块，其中最具特点的就是 selenium 模块。该模块是浏览器自动化测试框架，是一款用于 Web 应用程序测试的工具，可以直接运行在浏览器当中，如同用户在浏览器中操作一样。该工具所支持的浏览器有 IE、Mozilla Firefox 及 Google Chrome 等。

22.5.1 selenium 模块的安装

在命令行窗口中输入 pip install selenium 命令，然后按下<Enter>键，将显示如图 22.7 所示的安装进度。

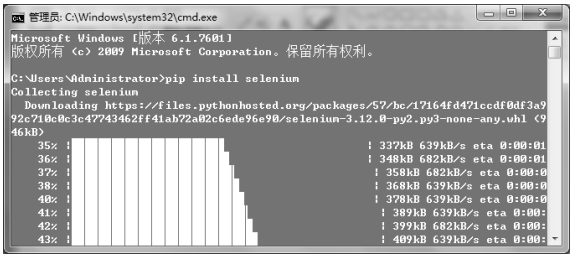


图 22.7　安装 selenium 模块

22.5.2 下载浏览器驱动

selenium 模块安装完成以后，还需要选择一个浏览器，然后下载对应的浏览器驱动，此时才可以通过 selenium 模块来控制浏览器的操作。这里选择 Google Chrome 66.0.3359.139（正式版本）浏览器，然后在谷歌浏览器驱动地址（http://chromedriver. storage.googleapis.com/index.html?path=2.38/）中下载 2.38 版本的浏览器驱动，如图 22.8 所示。

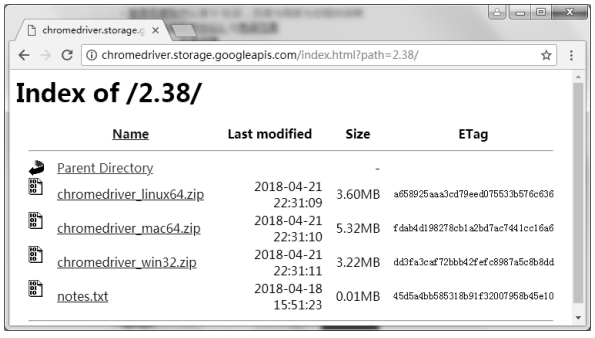


图 22.8　下载谷歌浏览器驱动

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

> **说明：**在下载谷歌浏览器驱动时，需要根据自己计算机的系统版本下载对应的浏览器驱动。这里以 Windows 系统为例，所以下载 chromedriver_win32.zip 即可。

22.5.3　selenium 模块的使用

谷歌浏览器驱动下载完成后，将名为 chromedriver.exe 的文件提取出来，保存在指定的路径当中，然后需要通过 Python 代码进行谷歌浏览器驱动的加载，这样才可以启动浏览器驱动并控制浏览器。下面通过一个示例实现启动谷歌浏览器并加载指定的网页，代码如下：

```
from selenium import webdriver # 导入浏览器驱动模块

try:
    # 创建谷歌浏览器驱动参数对象
    chrome_options = webdriver.ChromeOptions()
    # 加载谷歌浏览器驱动
    driver = webdriver.Chrome(chrome_options=chrome_options,executable_path=
'C:/Users/Administrator/AppData/Local/Programs/Python/chromedriver.exe')
    # 设置超时时间
    driver.set_page_load_timeout(10)
    # 请求地址
    driver.get('https://www.baidu.com/')
    driver.quit()  # 退出浏览器驱动
except Exception as e:
    print('显示异常信息！', e)
```

该示例运行完成后，将自动启动谷歌浏览器，然后自动加载指定的网页，加载完成后自动关闭浏览器。

22.5.4　selenium 模块的常用方法

selenium 模块不仅可以自动启动浏览器与加载指定的网页，还可以抓取网页中的内容，其中比较常用的方法如表 22.1 所示。

表 22.1　selenium 模块抓取网页内容的常用方法

常用方法	描　　述
element = driver.find_element_by_id("**coolestWidgetEvah**") 或 **from** selenium.webdriver.common.by **import** By element = driver.find_element(by=By.ID, value="**coolestWidgetEvah**")	根据 ID 找到如下所示的元素： ...

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

续表

常用方法	描　述
cheeses = driver.find_elements_by_class_name("**cheese**") 或 **from** selenium.webdriver.common.by **import** By cheeses = driver.find_elements(By.CLASS_NAME, "**cheese**")	根据类名找到如下所示的元素： Cheddar <divclass="cheese">Gouda
frame = driver.find_element_by_tag_name("**iframe**") 或 **from** selenium.webdriver.common.by **import** By frame = driver.find_element(By.TAG_NAME, "**iframe**")	根据标签名找到如下所示的元素： <iframe src="..."></iframe>
cheese = driver.find_element_by_name("**cheese**") 或 **from** selenium.webdriver.common.by **import** By cheese = driver.find_element(By.NAME, "**cheese**")	根据名称找到如下所示的元素： <input name="cheese" type="text"/>
cheese = driver.find_element_by_link_text("**cheese**") 或 **from** selenium.webdriver.common.by **import** By cheese = driver.find_element(By.LINK_TEXT, "**cheese**")	根据链接文本找到如下所示的元素： cheese >
cheese = driver.find_element_by_partial_link_text("**cheese**") 或 **from** selenium.webdriver.common.by **import** By cheese = driver.find_element(By.PARTIAL_LINK_TEXT, "**cheese**")	根据链接文字找到如下所示的元素： search for cheese>
cheese = driver.find_element_by_css_selector("**#food span.dairy.aged**") 或 **from** selenium.webdriver.common.by **import** By cheese = driver.find_element(By.CSS_SELECTOR, "**#food span.dairy.aged**")	通过 CSS 定位找到如下所示的元素： milk <span class= "dairy aged">cheese
element = driver.find_element_by_id("**element_id**") element.text	获取文本值

22.6　巡店宝（京东版）查询工具

22.6.1　窗体的实现

在创建巡店宝（京东版）查询工具的窗体时，需要考虑查询部分中包含用于输入图书名称的编辑框、用于查询的按钮、用于选择（好评、中评、差评）的单选按钮、用于显示图书信息部分、图书目录与好评度分析的窗口。最后还需要添加显示评价信息的列表控件。实现的具体步骤如下。

（1）创建名为巡店宝（京东版）的文件夹，该文件夹用于保存京东图书信息查询工具的项目文件。然后在该文件夹中创建一个名为 image 的文件夹，用于保存背景图片。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（2）打开 Qt Designer 工具，首先将主窗体最大尺寸与最小尺寸设置为 793px×650px，然后在主窗体中添加一个 Widget 控件，需要距离主窗体顶部 120px，该控件用于充当查询部分的容器。在该 Widget 控件中依次添加 Label 控件，用于显示“输入商品名称：”文字；Line Edit 控件，用于输入书名的编辑框；Push Button 控件，用于“查询”按钮；3 个 Radio Button 控件，分别用于选择好评、中评及差评的单选按钮。窗体效果如图 22.9 所示。

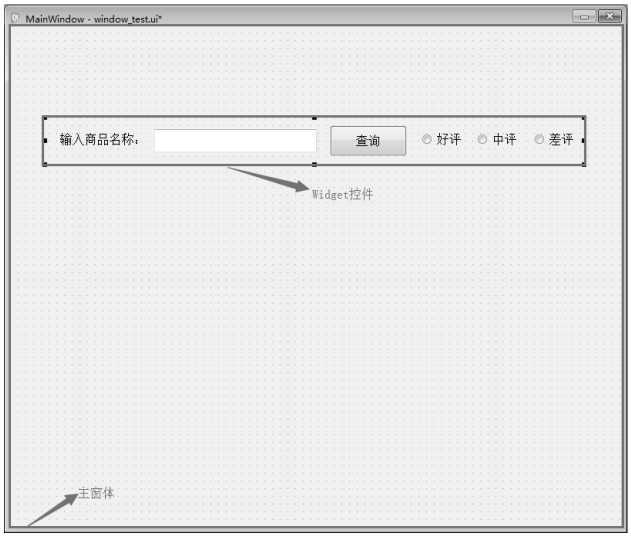


图 22.9　设计主窗体与查询部分

（3）在查询部分的下面添加 Widget 控件，用于充当显示部分的容器。在该步骤中主要使用 Label 控件实现图书信息的文字显示，如图 22.10 所示。

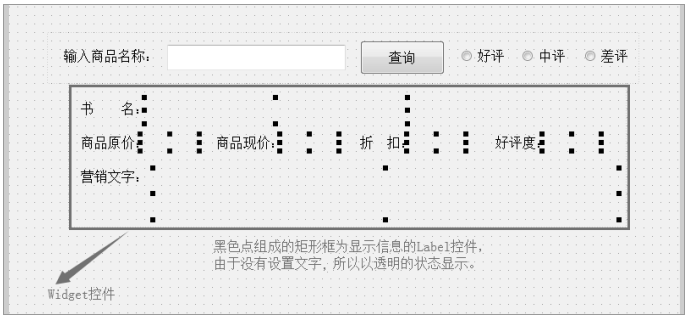


图 22.10　设计显示图书信息部分

（4）在显示图书信息部分的下面，首先添加显示排名信息的 Label 控件，然后添加“显示目录”按钮与“好评度分析”按钮，在该步骤中没有添加 Widget 控件，如图 22.11 所示。

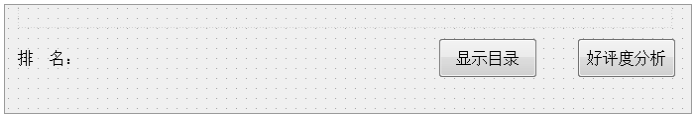


图 22.11　设计显示排名信息与“显示目录”和“好评度分析”按钮

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（5）在显示排名信息位置的下面，添加 List View 控件，用于显示评价信息，如图 22.12 所示。

图 22.12　设计显示评价信息的列表控件

（6）窗体设计完成以后，保存为 window.ui 文件，然后将该文件转换为 window.py 文件。转换完成以后，打开 window.py 文件，在 setupUi()方法中定义显示信息时的文字样式，并且将显示图书信息的 Label 控件使用该文字样式。代码如下：

```
self.textStyle = "QLabel{background:white;}""" \
                 "QLabel{color:rgb(231,40,59,250);font-weight:bold;}"
```

> **说明：**显示图书信息的 Label 控件在使用文字样式时，可以使用如下代码：
> `self.label_15.setStyleSheet(self.textStyle)`

（7）在设置主窗体代码的下面，设置主窗体背景图片。代码如下：

```
window_pale = QtGui.QPalette()  # 调色板类
# 设置背景图片
window_pale.setBrush(MainWindow.backgroundRole(),
                         QtGui.QBrush(QtGui.QPixmap("image/bg.png")))
MainWindow.setPalette(window_pale) # 为主窗体设置对应的调色板
```

（8）在显示部分 Widget 控件代码的下面，设置显示区域的背景图片。代码如下：

```
self.widget_2.setAutoFillBackground(True)  # 开启自动填充背景
palette = QtGui.QPalette()  # 调色板类
palette.setBrush(QtGui.QPalette.Background,
                 QtGui.QBrush(QtGui.QPixmap('image/show_bg.png')))  # 设置背景图片
self.widget_2.setPalette(palette)  # 为控件设置对应的调色板
```

（9）导入 sys 模块，然后在代码块的最外层创建 show_MainWindow()方法，用于显示窗体。关键代码如下：

```
import  sys
def show_MainWindow():
    app = QtWidgets.QApplication(sys.argv) # 实例化 QApplication 类，作为 GUI 主程序入口
    MainWindow = QtWidgets.QMainWindow()    # 创建 MainWindow
    ui = Ui_MainWindow()    # 实例化 UI 类
    ui.setupUi(MainWindow)  # 设置窗体 UI
    MainWindow.show()       # 显示窗体
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
08      sys.exit(app.exec_())  # 窗口创建完成，需要结束主循环过程
```

（10）在代码块的最外层模拟 Python 的程序入口，然后调用显示窗体的 show_MainWindow() 方法。关键代码如下：

```
01  if __name__ == "__main__":
02      show_MainWindow()
```

在该文件中单击鼠标右键，在弹出的快捷菜单中选择 Run 'window'命令，将显示如图 22.13 所示的主窗体界面。

图 22.13　主窗体界面

22.6.2　获取京东商品 name_id

经过分析与观察可以发现，在京东商城中打开每种商品的网页时，地址中都会有一串数字，而每个商品地址中的数字都是不同的。此时可以判断每种商品都有对应的 ID，而这个 ID 就是一串数字，如图 22.14 和图 22.15 所示。

图 22.14　《零基础学 Python》商品 ID

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 22.15 《零基础学 Java》商品 ID

发现此规律以后，在巡店宝（京东版）的文件夹中创建 name_id.py 文件，然后在该文件中创建一个名为 data_dict 的字典，用于保存图书名称与对应的 ID。具体代码如下：

```
data_dict = {
    # 零基础学系列
    '零基础学 Java': '12185501',
    '零基础学 Android': '12199033',
    '零基础学 C 语言': '12250414',
    '零基础学 C#': '12271986',
    '零基础学 Oracle': '12271911',
    '零基础学 JavaScript': '12275164',
    '零基础学 PHP': '12206201',
    '零基础学 HTML5+CSS3': '12271813',
    '零基础学 Python': '12353915',
    # 项目开发实战入门系列
    'Java 项目开发实战入门': '12163091',
    'Java Web 项目开发实战入门': '12163151',
    'JSP 项目开发实战入门': '12233850',
    'Android 项目开发实战入门': '12163105',
    'C 语言项目开发实战入门': '12163145',
    'C++项目开发实战入门': '12163147',
    'C#项目开发实战入门': '12163129',
    'ASP.NET 项目开发实战入门': '12163089',
    'PHP 项目开发实战入门': '12163093',
    # 200 例系列
    'Java 精彩编程 200 例': '12185937',
    'Android 精彩编程 200 例': '12237622',
    'C#精彩编程 200 例': '12235813',
    'C 语言精彩编程 200 例': '12199075',
    'Visual Basic 精彩编程 200 例': '12271763',
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
28      'SQL 即查即用': '12359944'
29  }
```

在获取京东商品 ID 的同时，就可以分析出查询商品信息的请求地址。从图 22.14 和图 22.15 中可以看出，对于不同的商品名称，只要将固定的地址更改为商品对应的 ID 即可。分析后，查询商品信息的请求地址如下：

```
'https://item.jd.com/{}.html'.format(商品对应的 ID)
```

> **注意：**由于京东官方网站不断更新，所以本项目实战中的所有信息均以京东官方网站更新后的为准！

22.6.3　获取京东商品 json_id

获取到京东商品 ID 以后，接下来需要考虑获取京东商品 json_id，该 ID 是获取商品评价时所使用的参数之一。获取京东商品 json_id 的具体步骤如下。

（1）在京东官方网站中，使用 Google Chrome 浏览器打开指定商品的网页，然后单击网页右侧的累计评价数，如图 22.16 所示。

图 22.16　单击累计评价数

（2）在显示的商品评价处，首先勾选“只看当前商品评价”复选框，然后在右侧的“推荐排序”中选择“时间排序”，如图 22.17 所示。

图 22.17　选择评价的显示方式

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（3）按快捷键<F12>打开开发者工具，先单击“Network”，再在网页中单击“好评”，此时在网络监控中将显示当前所触发的网络请求信息，如图 22.18 所示。

图 22.18　显示好评网络请求信息

（4）依次选择图 22.18 中的请求信息，然后单击“Preview”，可以查看请求后返回的结果预览，预览后发现第三条请求信息中返回的是一条 JSON 信息，如图 22.19 所示。

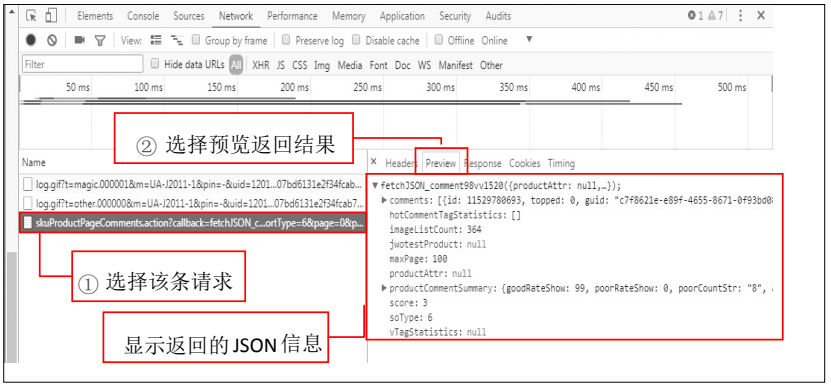


图 22.19　预览返回的 JSON 信息

（5）接下来需要判断该条 JSON 信息有没有可用数据，也就是确定数据中是否含有好评的信息文字。在预览的 JSON 信息中依次展开“comments”→“0”，然后将展开后的 JSON 信息与网页中的评价信息进行核对，如图 22.20 所示。

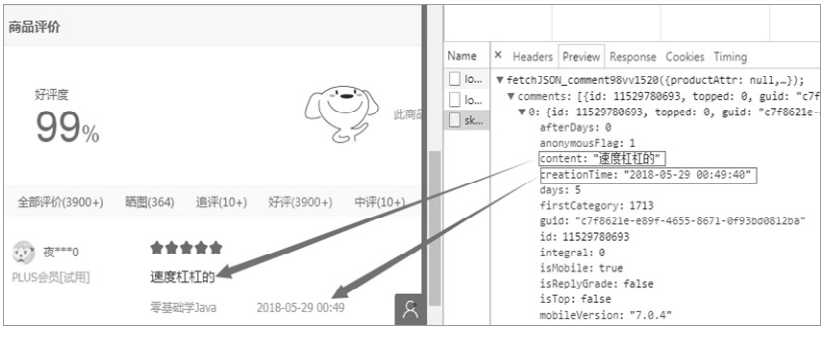


图 22.20　核对评价信息

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

说明：使用该方法在展开的 JSON 信息中可以依次找到用户的昵称、会员级别等信息。

（6）JSON 信息确认完成以后，选择“Headers”选项，查看好评请求地址及地址中所需的参数，如图 22.21 所示。

图 22.21　查看好评请求地址与请求参数

说明：请求参数中的每个参数都具有一定的含义。在发送好评信息的请求地址时，必须在请求地址中添加图 22.21 中比较重要的几个请求参数，否则无法获取返回的好评 JSON 信息。请求参数的含义如表 22.2 所示。

表 22.2　请求参数的含义

参数名称	含　义
callback	商品对应的 json_id
productId	商品对应的京东商品 ID
score	评价等级参数，差评为 1，中评为 2，好评为 3
sortType	排序的类型，6 为时间排序，5 为推荐排序
page	页数
pageSize	每页显示的评价信息数量

（7）了解好评请求参数后，可以知道 callback 参数名称所对应的值就是 json_id，它与京东商品 name_id 一样，每个商品都有自己的 json_id，此时可以根据以上方式获取其他商品的 json_id。然后在巡店宝（京东版）的文件夹中创建 json_id.py 文件，在该文件中创建一个名为 json_dict 的字典，用于保存图书名称与对应的 json_id。具体代码如下：

```
01 json_dict = {
02     # 零基础学系列
03     '零基础学 Java': 'fetchJSON_comment98vv1520',
04     '零基础学 Android': 'fetchJSON_comment98vv365',
05     '零基础学 C 语言': 'fetchJSON_comment98vv773',
06     '零基础学 C#': 'fetchJSON_comment98vv507',
07     '零基础学 Oracle': 'fetchJSON_comment98vv69',
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    '零基础学 JavaScript': 'fetchJSON_comment98vv267',
    '零基础学 PHP': 'fetchJSON_comment98vv373',
    '零基础学 HTML5+CSS3': 'fetchJSON_comment98vv175',
    '零基础学 Python': 'fetchJSON_comment98vv0',
    # 项目开发实战入门系列
    'Java 项目开发实战入门': 'fetchJSON_comment98vv506',
    'Java Web 项目开发实战入门': 'fetchJSON_comment98vv706',
    'JSP 项目开发实战入门': 'fetchJSON_comment98vv176',
    'Android 项目开发实战入门': 'fetchJSON_comment98vv375',
    'C 语言项目开发实战入门': 'fetchJSON_comment98vv427',
    'C++项目开发实战入门': 'fetchJSON_comment98vv520',
    'C#项目开发实战入门': 'fetchJSON_comment98vv527',
    'ASP.NET 项目开发实战入门': 'fetchJSON_comment98vv220',
    'PHP 项目开发实战入门': 'fetchJSON_comment98vv331',
    # 200 例系列
    'Java 精彩编程 200 例': 'fetchJSON_comment98vv733',
    'Android 精彩编程 200 例': 'fetchJSON_comment98vv578',
    'C#精彩编程 200 例': 'fetchJSON_comment98vv232',
    'C 语言精彩编程 200 例': 'fetchJSON_comment98vv265',
    'Visual Basic 精彩编程 200 例': 'fetchJSON_comment98vv56',
    'SQL 即查即用': 'fetchJSON_comment98vv1'
}
```

22.6.4 浏览器自动加载指定商品的网页

在实现浏览器自动加载指定商品的网页时，需要考虑到以下 3 点。

- 当浏览器自动加载网页信息时，是否需要加载图片。为了提高加载速度，建议选择不加载图片。
- 是否使用无界面浏览模式，让浏览器在后台自动加载；否则谷歌浏览器会自动弹出并启动加载指定的页面。
- 需要添加网页加载超时的异常提示。

实现的具体步骤如下。

（1）在巡店宝（京东版）的文件夹中创建 crawl.py 文件，然后在该文件中导入浏览器驱动模块、解析 HTML 模块及时间模块。代码如下：

```
from selenium import webdriver      # 浏览器驱动模块
from bs4 import BeautifulSoup       # 解析 HTML 模块
import time                         # 时间模块
```

（2）创建 Book 类，然后在该类中调用初始化 init()方法，并在该方法中进行初始化操作，保存评价信息的列表及网页超时标记。代码如下：

```
class Book(object):
    def __init__(self):
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
        self.evaluation_list = []        # 保存评价信息的列表
        self.is_overtime = None          # 网页超时标记
```

（3）顺序添加代码，创建 get_html()方法，该方法用于实现获取京东商品信息页面的 HTML 代码。首先需要设置浏览器驱动不加载图片及使用无界面浏览模式，然后通过谷歌浏览器驱动的绝对路径加载该驱动，再设置超时处理，如果没有超时，则说明网页加载成功，此时对网页中的 HTML 代码进行解析。具体代码如下：

```
# 获取 HTML 源码
def get_html(self, name_id, json_id):
    self.name_id = name_id
    self.json_id = json_id
    try:
        chrome_options = webdriver.ChromeOptions()
        # 不加载图片
        prefs = {"profile.managed_default_content_settings.images": 2}
        chrome_options.add_experimental_option("prefs", prefs)
        # 使用 headless 无界面浏览模式
        chrome_options.add_argument('--headless')
        chrome_options.add_argument('--disable-gpu')
        # 加载谷歌浏览器驱动，executable_path 为浏览器驱动的绝对路径
        driver = webdriver.Chrome(chrome_options=chrome_options,
                                  executable_path='D:/chromedriver.exe')
        # 设置超时时间
        driver.set_page_load_timeout(10)
        # 打开指定商品的网页
        driver.get('https://item.jd.com/{}.html'.format(self.name_id))
        print('请求完成')
        self.is_overtime = False     # 标记没有超时
    except Exception as e:
        print('网页加载超时！', e)
        driver.quit()                # 关闭浏览器驱动
        self.is_overtime = True      # 标记已经超时
    if self.is_overtime != True:     # 在网页加载没有超时的情况下
        time.sleep(3)                # 等待 3 秒
        html = driver.page_source    # 获取页面 HTML 代码
        self.html = BeautifulSoup(html, "html.parser")  # 解析 HTML 代码
        print('解析代码')
        print(self.html)             # 打印解析后的 HTML 代码
        driver.quit()                # 关闭浏览器驱动
```

（4）打开 window.py 文件，导入 name_id 与 json_id 文件及 Book 类，在 setupUi()方法中首先创建 Book 类的对象，然后定义网页信息是否显示完成的标记。代码如下：

```
self.book = Book()          # 创建 Book 类的对象
self.is_finish = False      # 定义网页信息是否显示完成的标记
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（5）在 retranslateUi()方法中设置排名文字的下面，为“查询”按钮指定单击事件。代码如下：

```
self.pushButton.clicked.connect(self.query_on_click) # 为“查询”按钮指定单击事件的方法
```

（6）在 retranslateUi()方法的下面创建 query_on_click()方法，该方法用于实现获取指定书名的网页信息。首先需要获取编辑框中所输入的书名，然后根据书名分别在 name_id 与 json_id 文件中找到对应的 ID，最后调用 Book 类中的 get_html()方法，并将获取的 name_id 与 json_id 作为该方法中的必要参数，完成图书网页的加载与 HTML 代码的解析工作。在 query_on_click()方法的下面创建 messageDialog()方法，用于显示消息提示的对话框。代码如下：

```
# “查询”按钮的单击事件
def query_on_click(self):
    self.input_book_name = self.lineEdit.text()         # 获取输入的书名
    if self.input_book_name != '':                      # 判断是否输入了书名
        if self.input_book_name in name_id.data_dict:   # 判断数据中是否有该书名
            # 获取书名对应的 ID
            self.name_id = name_id.data_dict[self.input_book_name]
            # 获取书名对应的评价 json_id
            self.json_id = json_id.json_dict[self.input_book_name]
            # 根据书名所对应的 ID 获取京东对应的 HTML 代码
            self.book.get_html(self.name_id, self.json_id)
        else:
            self.messageDialog('警告', '没有该书名！')
            print('没有该书名！')
    else:
        self.messageDialog('警告', '请输入需要查询的书名！')
        print('请输入需要查询的书名！')

# 提示对话框
def messageDialog(self, title, message):
     self.msg_box = QtWidgets.QMessageBox(QtWidgets.QMessageBox.Information,
                                                                  title, message)
     self.msg_box.exec_()
```

运行主窗体，然后输入需要查询的书名，单击“查询”按钮，（根据当前网络速度）等待一定的时间，在控制面板上将显示京东商品网页解析后的 HTML 代码，如图 22.22 所示。

```
<!--new book-->
<meta content="text/html; charset=utf-8" http-equiv="Content-Type"/>
<title>《零基础学Java（全彩版）（附光盘小白手册）》(明日科技)【摘要 书评 试读】- 京东图书</title>
<meta content="零基础学Java（全彩版）（附光盘小白手册）,,吉林大学出版社,9787569205688,,在线购买,折扣,打折" name="keywords"/>
<meta content="京东JD.COM图书频道为您提供《零基础学Java（全彩版）（附光盘小白手册）》在线选购，本书作者：，出版社：吉林大学出版社。
<meta content="telephone=no" name="format-detection"/>
<meta content="format=xhtml; url=//item.m.jd.com/product/12185501.html" http-equiv="mobile-agent"/>
<meta content="format=html5; url=//item.m.jd.com/product/12185501.html" http-equiv="mobile-agent"/>
```

图 22.22　显示京东商品网页解析后的 HTML 代码

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

22.6.5　分析与显示网页中的图书信息

在分析网页中的图书信息时，需要先在网页的 HTML 代码中找到需要提取的信息，然后观察该信息在 HTML 代码中的哪个标签下，以及标签中是否有可用的名称，例如，类名、标签名及 id 等。如果发现标签内的可用名称与其他多个标签内的可用名称相同，则还需要提取信息在多个标签中的具体位置，也可以理解为下标。例如，在当前页面中存在 5 个名为“item-mc”的类，而需要提取的信息在第 5 个包含“item-mc”类名的标签当中，则可以使用如下代码进行提取：

```
self.html.findAll('div', {'class', 'item-mc'})[4] # 第 5 个位置下标为 4
```

分析网页中的图书信息的具体步骤如下。

（1）首先使用 Google Chrome 浏览器在京东官方网站中打开指定商品的网页，然后按快捷键<F12>打开开发者工具，再单击 按钮，最后在网页中选中需要提取的信息。例如，要获取当前网页的标题名称，则可以按照如图 22.23 所示的步骤进行操作。

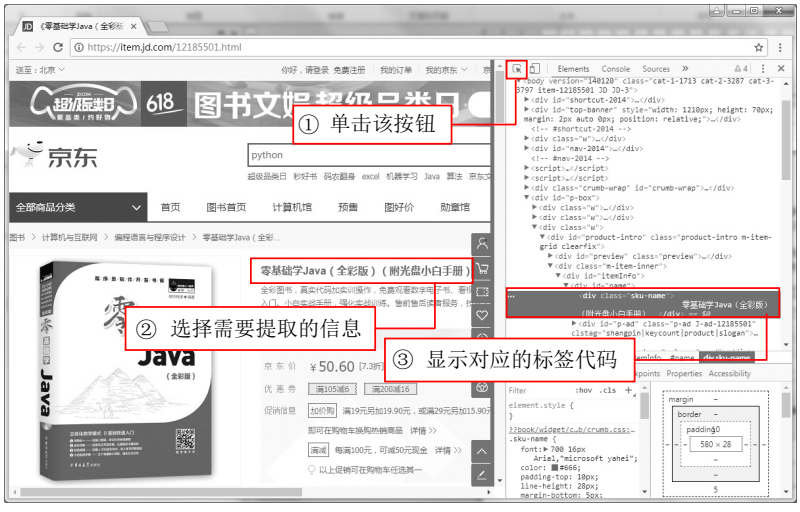


图 22.23　找到提取信息所对应的标签代码

（2）在对应的标签代码中找到标签名称，然后按快捷键<Ctrl+F>搜索对应的标签名称。如果在整个网页中只有一个标签名称，则直接获取该标签下的文字即可，如图 22.24 所示。

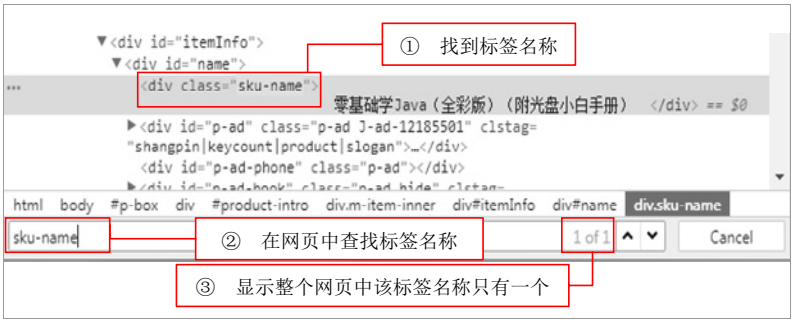


图 22.24　查找整个网页中标签名称的数量

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（3）确定了提取信息所对应的标签名称以后，打开 crawl.py 文件，然后在 get_html()方法的下面创建 get_title_name()方法，用于获取图书的标题名称。代码如下：

```
def get_title_name(self):
    # 在 HTML 文件中通过类名找到标题名称
    title_name = self.html.find('div', {'class', 'sku-name'})
    if title_name != None:                # 判断是否为空
        print(title_name.text.strip())
        return title_name.text.strip()   # 返回标题名称，并将左右两侧的空格去掉
    else:
        print('没有获取到标题名称！')
        return '没有获取到标题名称！'       # 返回提示信息
```

（4）重复步骤（1）和（2），通过浏览器开发者工具在 HTML 代码中找到商品现在的价格，如图 22.25 所示。

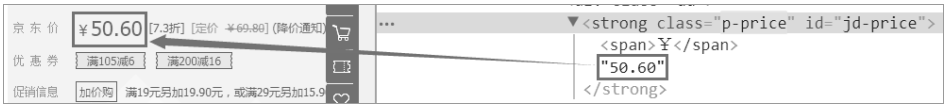


图 22.25　在 HTML 代码中找到商品现在的价格

（5）打开 crawl.py 文件，然后在 get_title_name()方法的下面创建 get_jd_price()方法，用于获取商品现在的价格。由于在 strong 标签中出现了两个可用的名称，分别是类名与 ID 名称，通过在整个网页中查找，发现 ID 名称是唯一的，所以商品现在价格的信息可以通过 ID 名称进行获取。代码如下：

```
def get_jd_price(self):
    # 在 HTML 文件中根据 ID 名称找到商品现在的价格
    jd_price = self.html.find('strong', id='jd-price')
    if jd_price != None:              # 判断是否为空
        print(jd_price.text)
        return jd_price.text          # 返回商品现在的价格
    else:
        print('没有获取到现在价格！')
        return '没有获取到现在价格！'   # 返回提示信息
```

（6）通过浏览器开发者工具在 HTML 代码中找到当前商品的折扣信息，如图 22.26 所示。

图 22.26　在 HTML 代码中找到当前商品的折扣信息

（7）打开 crawl.py 文件，然后在 get_jd_price()方法的下面创建 get_p_discount()方法，用于获取商品的折扣信息。代码如下：

```
def get_p_discount(self):
    # 在 HTML 文件中根据类名找到商品的折扣信息
    p_discount = self.html.find('span', {'class', 'p-discount'})
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
    if p_discount != None:          # 判断是否为空
        print(p_discount.text)
        return p_discount.text      # 返回折扣信息
    else:
        print('没有获取到折扣! ')
        return '没有获取到折扣! '    # 返回提示信息
```

（8）通过浏览器开发者工具在 HTML 代码中找到商品原价信息。由于在京东商城中经常会出现秒杀活动，经过观察发现，当出现秒杀活动时，原价信息所对应的标签与普通原价信息所对应的标签不同，如图 22.27 和图 22.28 所示。

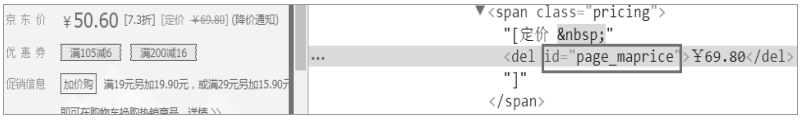


图 22.27　普通原价信息所对应的标签

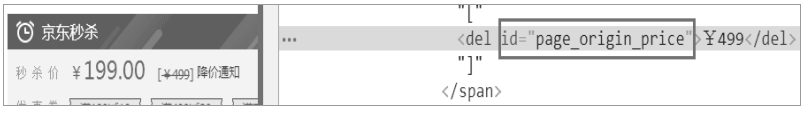


图 22.28　当出现秒杀活动时原价信息所对应的标签

> **说明：**从图 22.27 和图 22.28 中可以看出两者原价信息所对应标签中的 ID 名称不同，所以在获取商品原价信息时需要判断商品是否参与了秒杀活动。

（9）打开 crawl.py 文件，然后在 get_p_discount()方法的下面创建 get_page_maprice()方法，用于获取商品的原价信息。代码如下：

```
def get_page_maprice(self):
    # 在 HTML 文件中根据 ID 名称找到商品的原价信息
    page_maprice = self.html.find('del', id='page_maprice')
    if page_maprice != None:                # 判断是否为空
        print(page_maprice.text)
        return page_maprice.text            # 返回原价信息
    else:
        # 在 HTML 文件中根据 ID 名称找到商品在秒杀活动中的原价
        page_origin_price = self.html.find('del', id='page_origin_price')
        if page_origin_price != None:       # 判断是否为空
            print(page_origin_price.text)
            return page_origin_price.text   # 返回秒杀活动中的原价信息
        else:
            print('无秒杀价! ')
            return '无秒杀价! '              # 返回提示信息
        print('没有获取到原价! ')
        return '没有获取到原价! '
```

（10）通过浏览器开发者工具在 HTML 代码中找到当前商品的排名信息，有时会发现，在

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

提取信息的标签中包含不需要的信息，此时需要剥去外层标签，然后在子标签中提取自己需要的信息，如图 22.29 所示。

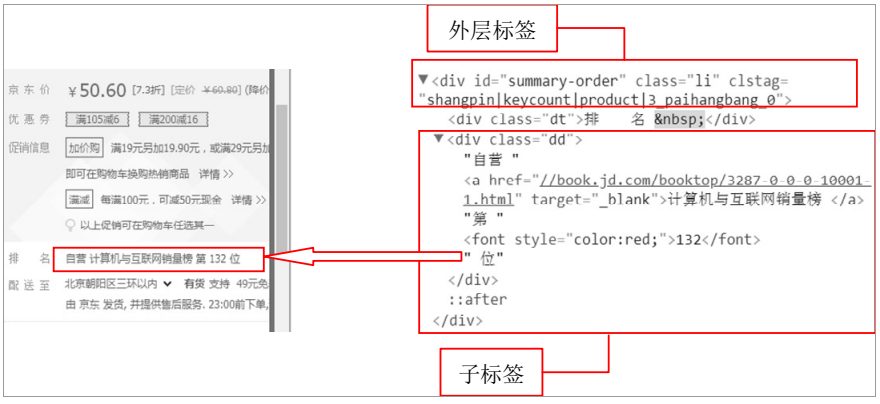


图 22.29　外层标签与子标签

（11）由于在京东商品中，不是每本书都会有排名信息的，所以需要在没有排名信息时给出提示。打开 crawl.py 文件，然后在 get_page_maprice()方法的下面创建 get_ranking()方法，用于获取排名信息。代码如下：

```
def get_ranking(self):
    # 在 HTML 文件中根据 ID 名称找到商品排名的外层标签
    ranking = self.html.find('div', id='summary-order')
    if ranking != None:  # 判断排名外层标签是否为空
        print(ranking.find('div', {'class', 'dd'}).text)
        return ranking.find('div', {'class', 'dd'}).text  # 返回获取的排名信息
    else:
        print('没有获取到排名！')
        return '没有获取到排名！' # 返回提示信息
```

（12）通过浏览器开发者工具在 HTML 代码中找到当前商品标题下的红色营销文字。仔细观察会发现，在获取不同商品的营销文字时，营销文字对应标签内的类名中包含该商品的 name_id，如图 22.30 所示。

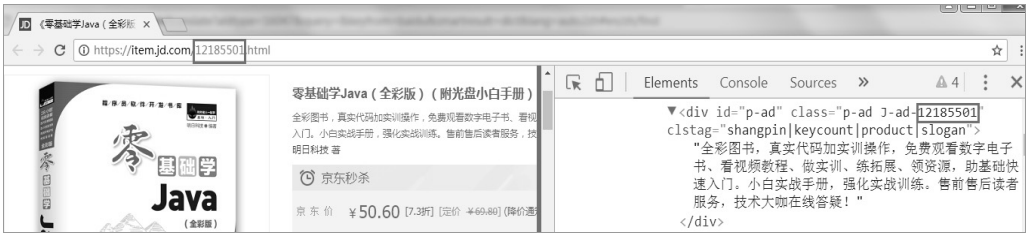


图 22.30　在 HTML 代码中找到商品营销文字

> **说明：**打开其他商品网页，在 HTML 代码中找到商品营销文字，确认是否与图 22.30 所示的营销文字相同，然后可以确认在 HTML 代码中获取营销文字时固定的类名称，需要将 name_id 更换为与商品名称对应的 name_id。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

（13）打开 crawl.py 文件，然后在 get_ranking()方法的下面创建 get_advertising()方法，用于获取商品的营销文字。代码如下：

```
def get_advertising(self):
    # 在 HTML 文件中根据类名找到商品营销文字，
    # 其中，将 name_id 设置为动态，实现在查询不同图书时获取对应的 name_id
    advertising = self.html.find('div', {'class', 'p-ad J-ad-{}'.format
(self.name_id)})
    if advertising != None:          # 判断是否为空
        print(advertising.text)
        return advertising.text      # 返回营销文字
    else:
        print('没有获取到营销文字！')
        return '没有获取到营销文字！'   # 返回提示信息
```

（14）打开 window.py 文件，然后在 messageDialog()方法的下面创建 show_message()方法，用于显示获取的图书信息。代码如下：

```
def show_message(self):
    # 显示书名信息
    self.label_15.setText(self.book.get_title_name())
    # 显示现在价格
    self.label_8.setText(self.book.get_jd_price())
    # 显示原价
    self.label_7.setText(self.book.get_page_maprice())
    # 显示折扣
    self.label_9.setText(self.book.get_p_discount())
    # 显示营销文字
    self.label_11.setText(self.book.get_advertising())
    # 显示排名
    self.label_13.setText(self.book.get_ranking())
```

（15）在 query_on_click()方法中 self.book.get_html(self.name_id, self.)代码的下面，判断网页请求是否超时，如果没有超时，则显示获取的信息。关键代码如下：

```
if self.book.is_overtime!=True:                    # 判断网页请求是否超时
    self.show_message()                            # 显示获取的信息
else:
    self.messageDialog('警告','网络异常或加载超时！请关闭应用重新查询！')
    print('网络异常或加载超时！')
```

运行主窗体，然后输入需要查询的书名，单击“查询”按钮，（根据当前网络速度）等待一定的时间，主窗体中将显示当前图书的相关信息，如图 22.31 所示。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

图 22.31　显示当前图书的相关信息

22.6.6　显示目录窗口

当实现显示目录窗口时，由于图书目录较长，所以本能反应一定是使用循环遍历的方式获取所有的目录。其实，在 Qt 窗体搭建工具中，提供了一个 QWebEngineView 控件，使用该控件可以显示本地的页面。此时可以将图书目录的 HTML 代码保存在本地文件中，然后通过窗体与 QWebEngineView 控件将图书的所有目录显示出来。显示目录窗口的具体步骤如下。

（1）在巡店宝（京东版）的文件夹中创建 file.py 文件，在该文件中首先导入系统操作模块 os；然后创建 write()方法，用于将图书目录部分的 HTML 代码写入文件；再创建 read()方法，用于读取目录文件；最后创建 is_html()方法，用于判断保存目录的 HTML 文件是否存在。具体代码如下：

```
01 import os  # 导入系统操作模块
02
03
04 def write(html, name):
05     file = open(name, 'w', encoding='utf_8_sig')  # 以写模式打开文件
06     file.write(html)         # 写入数据
07     file.close()
08
09 def read(name):
10     file = open(name, 'r', encoding='utf_8_sig')  # 以读模式打开文件
11     data = file.read()       # 读取文件
12     file.close()
13     return data
14
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
def is_html(name):
    is_html = os.path.exists(name)  # 判断 HTML 文件是否存在
    return is_html
```

（2）通过浏览器开发者工具在 HTML 代码中找到图书目录，经过分析可以看出，图书目录的标签中包含隐藏目录的样式代码，如图 22.32 所示。

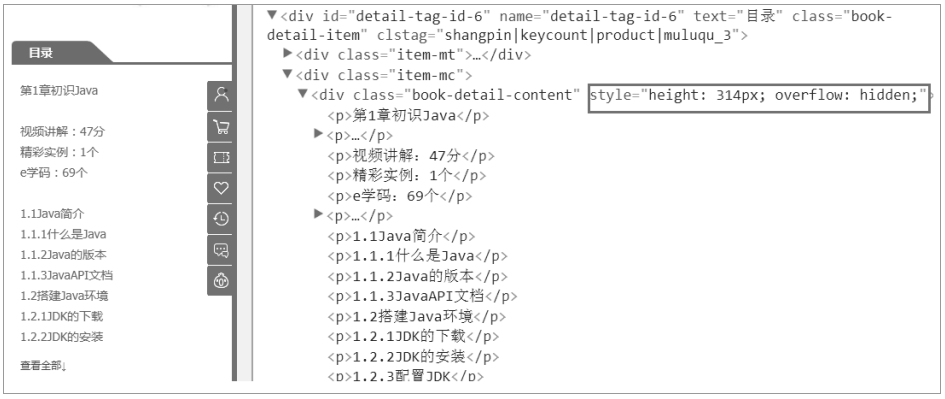


图 22.32　查找隐藏目录的样式代码

> **说明：**通过测试发现，如果删除方框内的样式代码，则在网页中将显示全部的目录信息。所以，此处需要考虑将样式代码删除后，再将图书目录的 HTML 代码保存在项目文件夹当中；否则在 QWebEngineView 控件中同样不会显示隐藏部分的目录信息。

（3）打开 crawl.py 文件，首先在顶部编写导入（from file import write）自定义的读写文件，然后在 get_advertising()方法的下面创建 get_catalog()方法，用于获取图书目录的信息，并将该信息通过写入文件的方式保存在项目文件夹[巡店宝（京东版）]当中。代码如下：

```
def get_catalog(self):
    # 根据 ID 名称找到外层标签
    catalog = self.html.find('div', id='detail-tag-id-6')
    if catalog != None: # 判断外层标签是否存在
        # 根据类名找到子标签
        catalog = catalog.find('div', {'class', 'book-detail-content'})
        if catalog != None: # 判断子标签是否存在
            # 将目录的 HTML 代码写成文件保存，并去除指定的样式代码
            write(str(catalog).replace(
                'style="height: 314px; overflow: hidden;"', ''),'catalog.text')
```

（4）打开 window.py 文件，然后在 show_message()方法中显示排名代码的下面调用 get_catalog()方法，实现获取目录信息并写成文件保存的功能，再标记为显示完成。代码如下：

```
self.book.get_catalog()
self.is_finish = True  # 显示完成
```

（5）在 window.py 文件中，首先在顶部编写导入（import file）自定义的读写文件，以及导入（from PyQt5.QtWebEngineWidgets import *）网页模块，然后在 Ui_MainWindow 类的下

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

面创建 CatalogWindow 类，用于初始化显示目录的窗体。代码如下：

```
class CatalogWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super(CatalogWindow, self).__init__()
        self.setWindowTitle('显示图书目录')        # 设置窗体标题名称
        self.setGeometry(5, 30, 600, 730)        # 设置窗体大小与位置
        self.browser = QWebEngineView()          # 创建加载 HTML 代码的控件
        # 加载 HTML 代码
        self.browser.setHtml(file.read('catalog.text'))
        self.setCentralWidget(self.browser)
```

（6）在 retranslateUi()方法中，为“显示目录”按钮指定单击事件。代码如下：

```
# 为“显示目录”按钮指定单击事件
self.pushButton_2.clicked.connect(self.catalog_on_click)
```

（7）在 show_message()方法的下面创建 catalog_on_click()方法，用于显示目录窗体与信息。代码如下：

```
def catalog_on_click(self):
    if file.is_html('catalog.text') and self.is_finish:
        self.win = CatalogWindow()  # 创建显示目录的窗体对象
        self.win.show()             # 显示目录窗体
    else:
        self.messageDialog('警告', '没有获取到目录文件或没有查询相关书籍的信息！')
        print('没有获取到目录文件或没有查询相关书籍的信息！')
```

运行主窗体，然后输入需要查询的书名，单击“查询”按钮，（根据当前网络速度）等待一定的时间，当主窗体中显示当前图书的相关信息以后，单击“显示目录”按钮，在屏幕的左上角将弹出显示图书目录的窗体，如图 22.33 所示。

图 22.33　显示图书目录的窗体

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

22.6.7　显示好评、中评及差评信息

因为在主窗体中已经设置了 3 个显示评价信息的单选按钮，所以可以通过这 3 个单选按钮实现发送不同的评价请求。例如，当选中“好评”单选按钮时，可以发送获取好评信息的请求。在 22.6.3 节中已经分析出获取好评信息的请求地址与参数，修改参数“score”就可以实现获取其他评价的请求信息。显示评价信息的具体步骤如下。

（1）打开 crawl.py 文件，首先导入（import json）解析 JSON 数据模块、（import re）正则表达式模块及（import requests）网络请求模块，然后在 get_catalog()方法的下面创建 get_evaluation()方法，用于实现发送评价请求并获取评价信息。在 get_evaluation()方法中首先定义好评率变量，然后清空评价数据列表，最后定义参数字典。代码如下：

```
# 获取评价内容，score 参数，差评为 1，中评为 2，好评为 3
def get_evaluation(self, score):
    # 好评率
    self.good_rate_show = None
    # 清空评价数据列表
    self.evaluation_list.clear()
    # 请求地址参数，callback 为书名 JSON 的 ID
    # productId 为书名对应的京东 ID
    # score 为评价等级参数，差评为 1，中评为 2，好评为 3
    # sortType 为排序的类型，6 为时间排序，5 为推荐排序
    # pageSize 为每页显示的评价数量
    # page 为页数
    params = {
        'callback': self.json_id,
        'productId': self.name_id,
        'score': score,
        'sortType': 6,
        'pageSize': 10,
        'isShadowSku': 0,
        'page': 0,
    }
```

（2）顺序添加代码，通过 while 循环发送评价请求，判断当页数大于 10 页时跳出循环，不再获取评价信息，最后通过 for 循环遍历评价中的每条信息，例如，时间、昵称等。代码如下：

```
    while True:
        # 评价请求地址
        url = 'https://club.jd.com/comment/skuProductPageComments.action'
        # 发送请求
        evaluation_response = requests.get(url, params=params)
        if evaluation_response.status_code == 200:
            evaluation_response = evaluation_response.text
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
            try:
                # 去除 JSON 外层的括号与名称
                t = re.search(r'({.*})', evaluation_response).group(0)
            except Exception as e:
                break
                print('评价的 JSON 数据匹配异常！ ')
            j = json.loads(t)  # 加载 JSON 数据
            # 获取评价信息内容
            commentSummary = j['comments']
            # 判断没有评价信息时显示无
            if len(commentSummary)==0:
                self.evaluation_list.append('无')
            # 获取好评率
            if self.good_rate_show==None:
                self.good_rate_show j['productCommentSummary']['goodRateShow']
                print('好评度',self.good_rate_show)
            # 当多于 10 页评价信息时跳出循环
            if params['page'] > 10:
                break
            for comment in commentSummary:
                # 评价内容
                c_contetn = comment['content']
                # 时间
                c_time = comment['creationTime']
                # 京东昵称
                c_name = comment['nickname']
                # 通过哪种平台购买
                c_client = comment['userClientShow']
                # 会员级别
                c_userLevelName = comment['userLevelName']
                # 差评为 1，中评为 2，好评为 3
                c_score = comment['score']
                # 定义显示格式并添加到保存评价信息的列表中
                self.evaluation_list.append(
                    '\n{} {} {} {}  {}\n{}\n'.format(
                        c_name, c_userLevelName, c_time, str(c_score)
                                                    + '颗星', c_client,c_contetn))
        else:
            print('评价请求失败！ ')
        params['page'] += 1 # 页数加 1
```

（3）打开 window.py 文件，首先使用 import threading 命令导入子线程模块，然后在 retranslateUi()方法中为评价单选按钮指定选中事件。代码如下：

```
self.radioButton.toggled.connect(lambda: self.btnstate(self.radioButton)) # 好评
self.radioButton_2.toggled.connect(lambda: self.btnstate(self.radioButton_2)) # 中评
```


電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
03  self.radioButton_3.toggled.connect(lambda: self.btnstate(self.radioButton_3)) # 差评
```

（4）在 query_on_click()方法中的 self.show_message()代码的下面，添加显示好评信息的代码。代码如下：

```
self.show_good_evaluation()                    # 显示好评信息
```

（5）在 show_message()方法中设置默认选中“好评”单选按钮。代码如下：

```
self.radioButton.setChecked(True)              # 设置默认选中“好评”单选按钮
```

（6）在 catalog_on_click()方法的下面创建 btnstate()方法，用于判断当前选中的是哪个单选按钮。代码如下：

```
01 # 评价按钮处理
02 def btnstate(self, btn):
03     if btn.text() == '好评':                  # 判断是不是“好评”单选按钮
04         if btn.isChecked() == True:           # 判断是否被选中
05             if self.is_finish == True:        # 判断图书信息是否显示完成
06                 # 通过线程获取好评信息
07                 th = threading.Thread(target=self.show_good_evaluation)
08                 th.start()
09             else:
10                 self.messageDialog('警告', '没有查询相关书籍的信息! ')
11                 print('没有查询相关书籍的信息! ')
12     if btn.text() == '中评':
13         if btn.isChecked() == True:
14             if self.is_finish == True:
15                 th = threading.Thread(target=self.show_general_evaluation)
16                 th.start()
17             else:
18                 self.messageDialog('警告', '没有查询相关书籍的信息! ')
19                 print('没有查询相关书籍的信息! ')
20     if btn.text() == '差评':
21         if btn.isChecked() == True:
22             if self.is_finish == True:
23                 th = threading.Thread(target=self.show_poor_evaluation)
24                 th.start()
25             else:
26                 self.messageDialog('警告', '没有查询相关书籍的信息! ')
27                 print('没有查询相关书籍的信息! ')
```

（7）在 btnstate()方法的下面，分别创建用于显示好评、中评、差评信息的 3 个方法。代码如下：

```
01 # 显示好评
02 def show_good_evaluation(self):
03     self.book.get_evaluation(3)     # 发送好评请求
04     self.set_evaluation_list()      # 设置好评数据
05     self.label_10.setText(str(self.book.good_rate_show) + '%') # 显示好评度
06
```

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

```
# 显示中评
def show_general_evaluation(self):
    self.book.get_evaluation(2)     # 发送中评请求
    self.set_evaluation_list()      # 设置中评数据

# 显示差评
def show_poor_evaluation(self):
    self.book.get_evaluation(1)     # 发送差评请求
    self.set_evaluation_list()      # 设置差评数据
```

（8）在 window.py 文件中导入（from PyQt5.QtCore import QStringListModel）字符串列表模式，然后为 List View 控件设置列表子项不可编辑属性，以及设置字符串列表模式。代码如下：

```
# 设置列表子项不可编辑
self.listView.setEditTriggers(QtWidgets.QListView.NoEditTriggers)
self.model = QStringListModel()  # 设置字符串列表模式
```

（9）在 show_poor_evaluation()方法的下面创建 set_evaluation_list()方法，用于设置评价列表中所显示的数据。代码如下：

```
def set_evaluation_list(self):
    # 获取评价数据
    self.list = self.book.evaluation_list
    # 设置列表显示内容
    self.model.setStringList(self.list)
    # 设置模式
    self.listView.setModel(self.model)
```

运行主窗体，然后输入需要查询的书名，单击“查询”按钮，（根据当前网络速度）等待一定的时间，主窗体中将显示当前图书的评价信息，如图 22.34 所示。

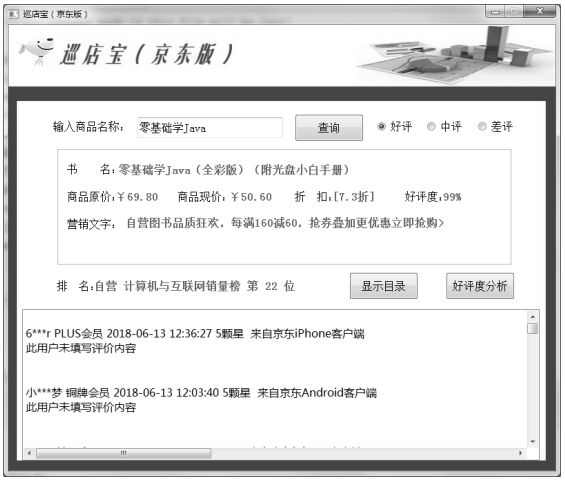


图 22.34　显示当前图书的评价信息

说明：默认显示好评信息，根据需求选中其他单选按钮将显示对应的评价信息。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

22.6.8　显示好评度分析窗口

当显示了评价信息以后，通过已经获取的好评度，即可制作一个用于显示好评度分析的可视化饼图窗口。具体步骤如下。

（1）在巡店宝（京东版）的文件夹中创建 charts.py 文件，在该文件中首先导入绘图模块，然后创建 Charts 类，在该类中创建 show_chart()方法，用于显示好评度分析的饼图窗口。代码如下：

```
01 import matplotlib.pyplot as plt  # 导入绘图模块
02 import matplotlib
03
04 """
05 绘制饼图
06 explode：设置各部分突出
07 label:设置各部分标签
08 labeldistance:设置标签文本与圆心的距离，1.1 表示 1.1 倍半径
09 autopct：设置圆里面的文本
10 shadow：设置是否有阴影
11 startangle：起始角度，默认从 0 开始逆时针转
12 pctdistance：设置圆内文本与圆心的距离
13 返回值
14 l_text：圆内部文本，matplotlib.text.Text object
15 p_text：圆外部文本
16 """
17 class Charts(object):
18     def __init__(self):
19         # 避免中文乱码
20         matplotlib.rcParams['font.sans-serif'] = ['SimHei']
21         matplotlib.rcParams['axes.unicode_minus'] = False
22
23     def show_chart(self, good_size, general_poor_size,  title):
24         label_list = ['好评', '中差评']           # 各部分标签
25         size = [good_size, general_poor_size]   # 各部分大小
26         color = ['lightblue',  'red']           # 各部分颜色
27         explode = [0.05, 0]                     # 各部分突出值
28         plt.pie(size, colors=color, labels=label_list, explode=explode,
29                 labeldistance=1.1,autopct="%1.1f%%", shadow=True,
30                 startangle=0, pctdistance=0.6)
31         plt.axis("equal")    # 设置横轴和纵轴大小相等，这样饼才是圆的
32         plt.title(title, fontsize=12)
33         plt.legend()         # 显示图例
34         plt.show()           # 显示绘图窗体
```

（2）打开 window.py 文件，首先导入（from charts import Charts）自定义显示好评度分析

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

的饼图窗口，然后在 retranslateUi()方法中为“好评度分析”按钮指定单击事件。代码如下：

```
# 为“好评度分析”按钮指定单击事件
self.pushButton_3.clicked.connect(self.analysis_on_click)
```

（3）在 set_evaluation_list()方法的下面创建 analysis_on_click()方法，用于实现单击“好评度分析”按钮后显示好评度分析的饼图窗口。代码如下：

```
01 # “好评度分析”按钮的单击事件
02 def analysis_on_click(self):
03     if self.is_finish:  # 判断图书信息标记
04         ch = Charts()    # 创建显示好评度分析的饼图窗口对象
05         # 显示窗口
06         ch.show_chart(self.book.good_rate_show, (100 self.book.good_rate_show),
07                       self.label_15.text())
08     else:
09         self.messageDialog('警告', '没有查询相关书籍的信息！')
10         print('没有查询相关书籍的信息！')
```

运行主窗体，然后输入需要查询的书名，单击“查询”按钮，（根据当前网络速度）等待一定的时间，当主窗体中显示当前图书的相关信息以后，单击“好评度分析”按钮，将显示好评度分析的饼图窗口，如图 22.35 所示。

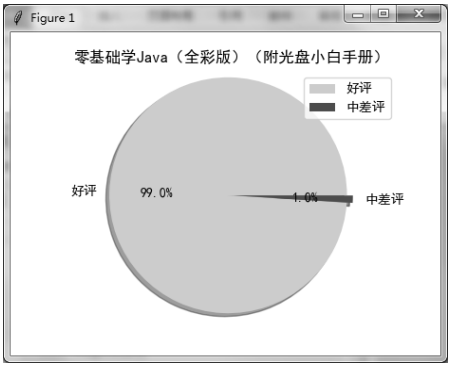


图 22.35　显示好评度分析的饼图窗口

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

反侵权盗版声明

电子工业出版社依法对本作品享有专有出版权。任何未经权利人书面许可，复制、销售或通过信息网络传播本作品的行为；歪曲、篡改、剽窃本作品的行为，均违反《中华人民共和国著作权法》，其行为人应承担相应的民事责任和行政责任，构成犯罪的，将被依法追究刑事责任。

为了维护市场秩序，保护权利人的合法权益，我社将依法查处和打击侵权盗版的单位和个人。欢迎社会各界人士积极举报侵权盗版行为，本社将奖励举报有功人员，并保证举报人的信息不被泄露。

举报电话：（010）88254396；（010）88258888

传　　真：（010）88254397

E－mail：　dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱　电子工业出版社总编办公室

邮　　编：100036

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY